**Bank management system**



Session: 2022 – 2026

**Submitted by:**

M. Abdullah Iftikhar 2022-CS-35

**Supervised by:**

Maida Shahid

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Contents**

Project Descriptions ……………………………………………. 3

Users of Application …………………………………………… 4

Functional Requirements ………………………………………. 5

OOP Concepts……………………………………………………6

CRC Diagram ....………………………………………………... 7

Complete Code …………………………………………………..8

**You Tube Link**

[**https://youtu.be/IqeITauGvXM**](https://youtu.be/IqeITauGvXM)

#  Short Description of my project

• Bank management system will keep financial records of all the people affiliated with the bank, e.g.,, and it works with arrow keys,

1. keep record of total available amount,
2. Remaining amount after withdrawal,
3. Allow them to take loans (for a specific period of time),
4. Help a group of people to start a committee, etc.
5. Pay Bills
6. Collect funds
7. Keep their money safe (since only one user has access to an account)

#  Users of Application

• Manager: Who specifically control what the clients or Employees will access &

do

• Costumer / Client: people who are affiliated with the Bank

#  Functional Requirements

***User As a I can So that I can***

***Story***

***ID***

|  |  |  |
| --- | --- | --- |
|  | Login to my account  Add an account  Delete an account  **After Login**  Confirm Addition of any new client  Remove an existing client if need be  Check requests (for loans, new cheque book, new card etc.)  Check client details (account balance, transection history etc.)  Check Complaints  Check Transaction History (For Whole Day, past week, past month)  Check Due Loans  Check Employee Details (vacancies, salary, leaves, etc.)  Check Funds (for hospital, orphanage,  Check available Balance in Bank |  |
| Client | Login to my account  Add an account  Delete an account  **After Login**  Check my Balance  Withdraw Cash  Deposit Cash  Request for a Loan  Pay Loan  Send Cash to another user  Pay bills (electricity, Gas, School etc.)  File a Complaint  Change pin  Transection history  Other services (request for new card, cheque book, etc.)  Log out |  |

**OOP Concepts:**

There is inheritance between users, in which PersonClass is the parent class and CdminClass(manager) and ClientClass (account holders). Users have same attributes .of name, password and role,  
 There is polymorphism in many functions as one class of person is made which leads toward the need to override some functions

There is also inheritance in loanClass, in which you are charged with interests depending on the province of your domicile

So main polymorphed function is calculate interest,

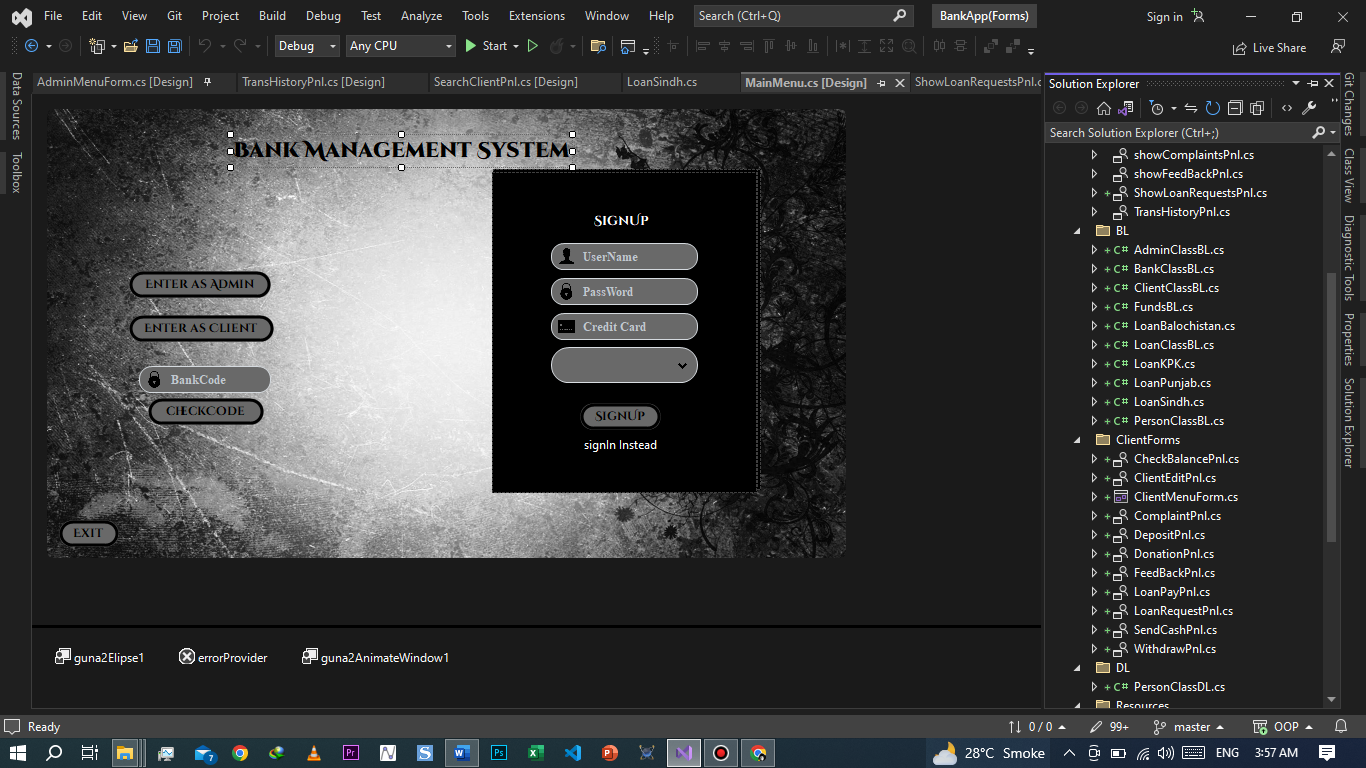
.

Data is being stored in file properly, in a single file

Funds and LoanClass are aggregated to client class in 1 to 1 relation.

PersonClass and personClassDL has 1 to many relation in aggregation.

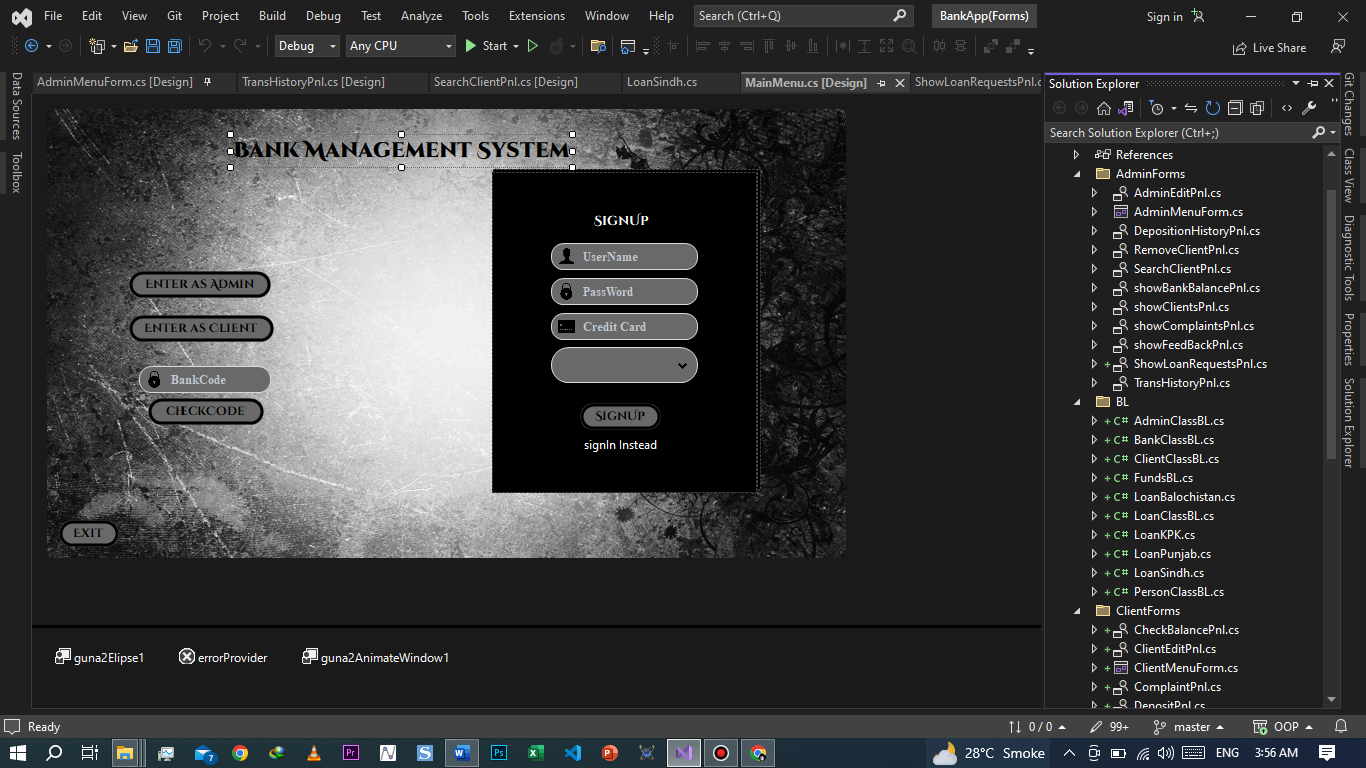
By using these concepts, it become very easy to manage the code. It helps to solve privacy issue as well as management issues.

**Design Pattern Implementation:**

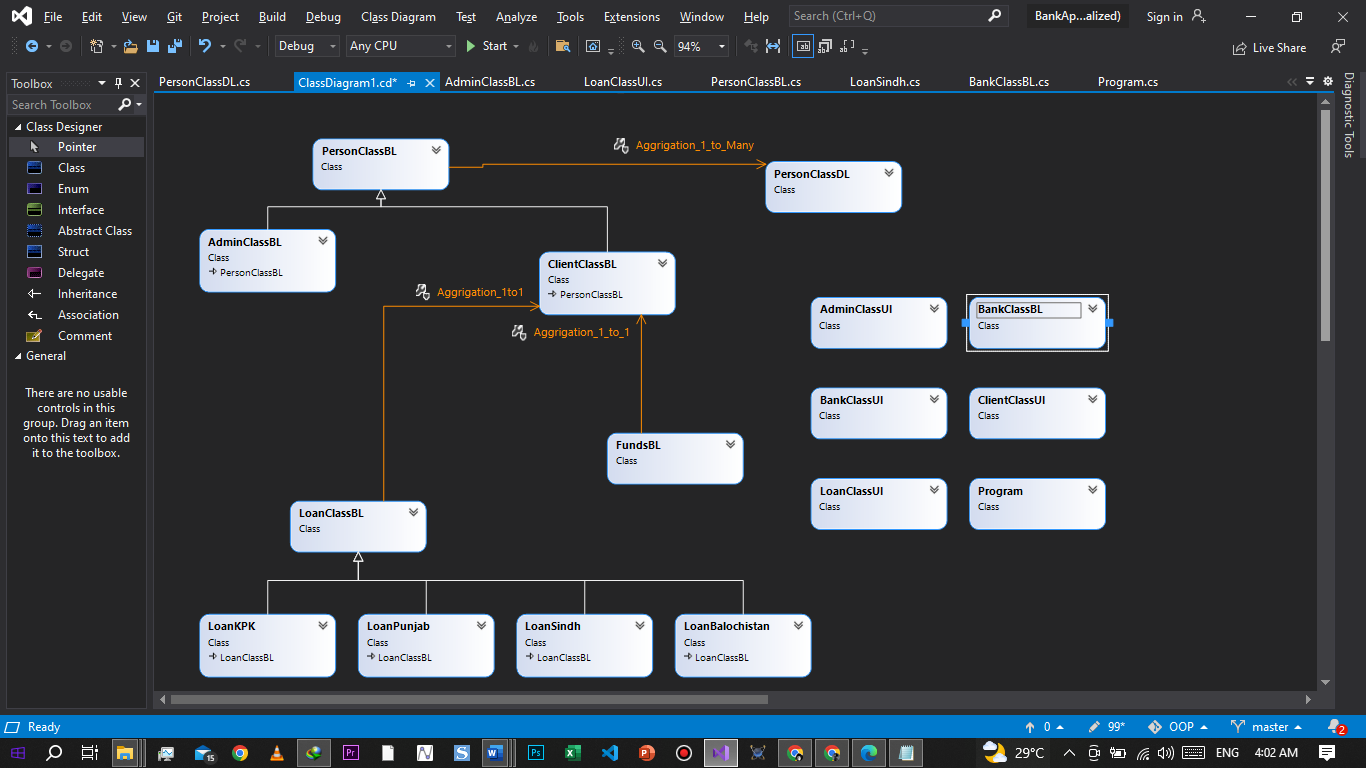
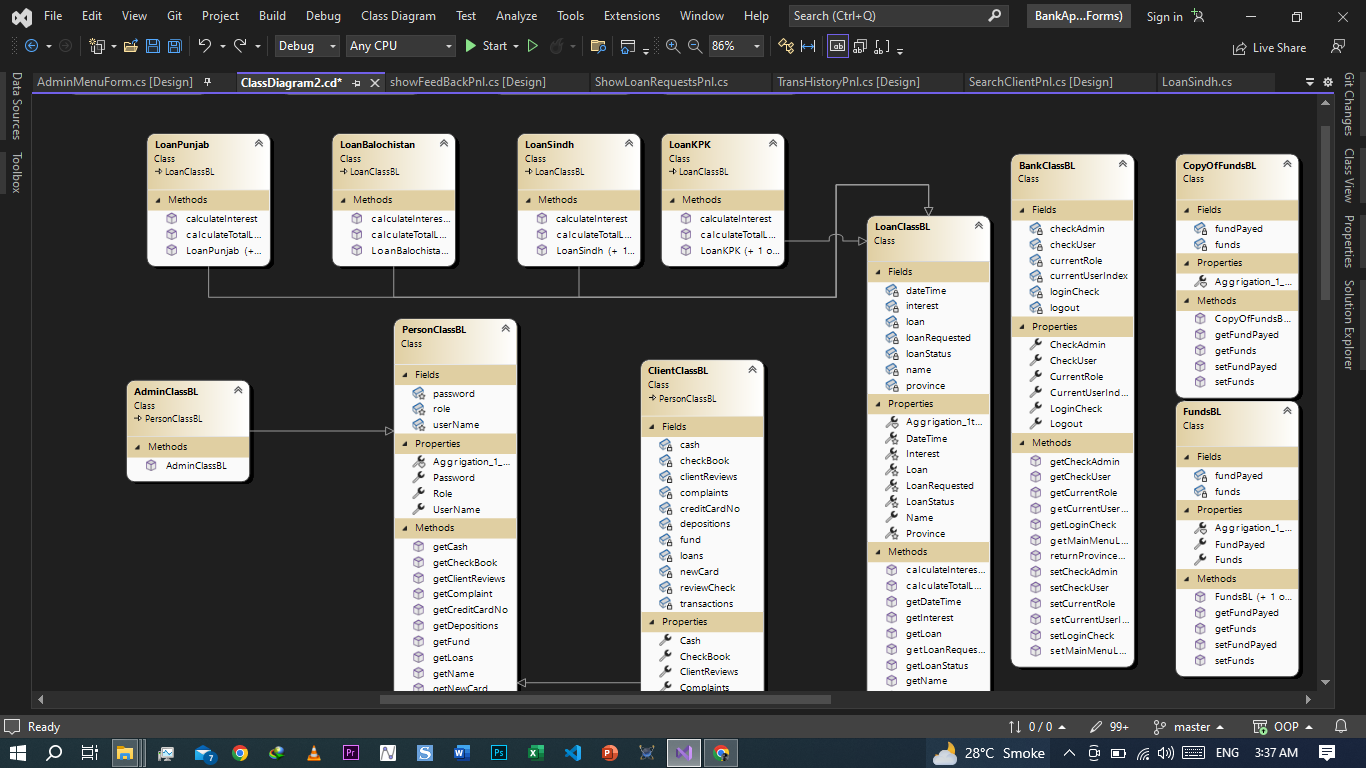
Project has following classes

Including 10 Classes of business layers  
1 Data Layer

And GUI made by panels

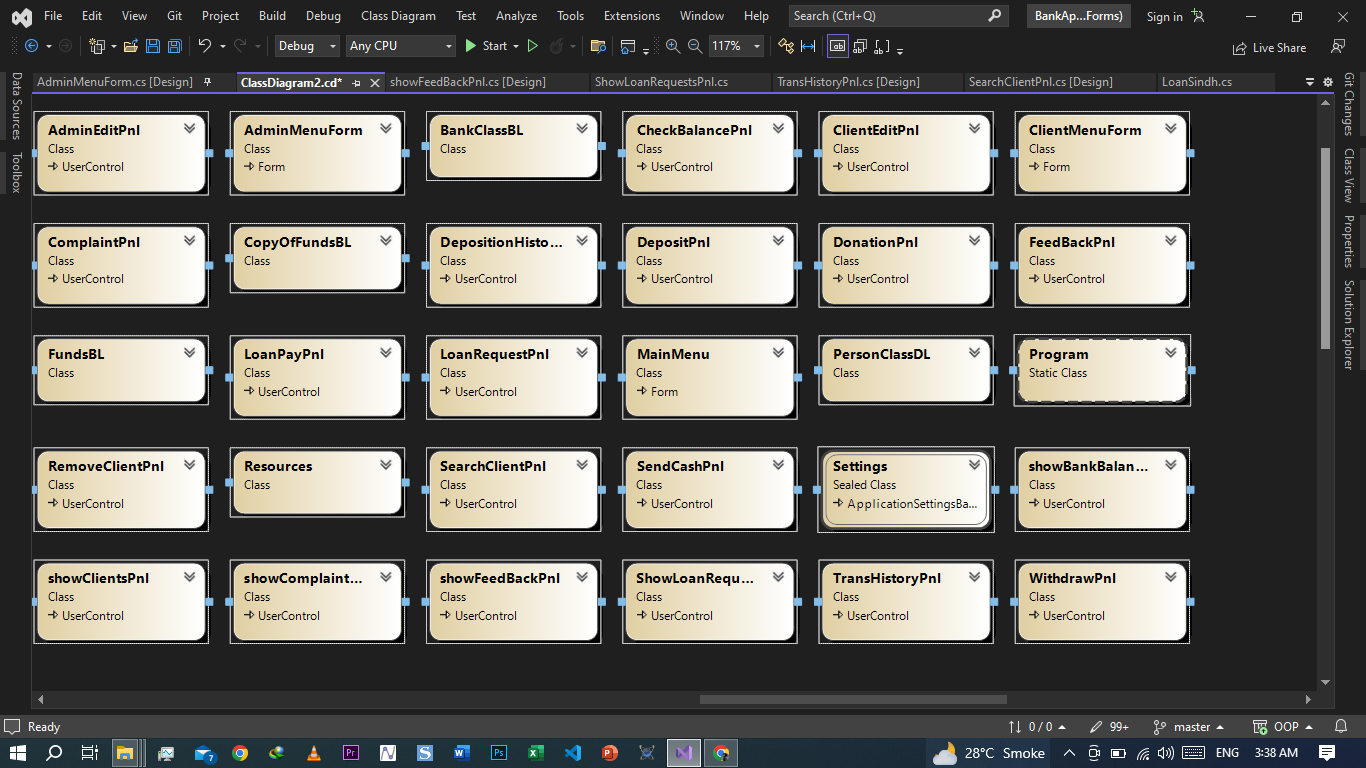


**CRC Diagram:**



**Class Details:**

PersonClassBL contains common attributes(name, password and role) of ClientClassBL and AdminClassBL,, ClientClassBL has further attributes like loans ,funds,balance etc,, which AdminClassBL doesn’t have.



**Complete Code**

**BL Classes**

**BankClassBL**

internal class BankClassBL

{

private static int currentUserIndex;

private static bool checkAdmin;

private static bool loginCheck;

private static bool checkUser;

private static bool logout;

private static string currentRole;

public static int CurrentUserIndex { get => currentUserIndex; set => currentUserIndex = value; }

public static bool CheckAdmin { get => checkAdmin; set => checkAdmin = value; }

public static bool LoginCheck { get => loginCheck; set => loginCheck = value; }

public static bool CheckUser { get => checkUser; set => checkUser = value; }

public static bool Logout { get => logout; set => logout = value; }

public static string CurrentRole { get => currentRole; set => currentRole = value; }

public static int getCurrentUserIndex()

{

return CurrentUserIndex;

}

public static void setCurrentUserIndex(int index)

{

BankClassBL.CurrentUserIndex = index;

}

public static bool getCheckAdmin()

{

return checkAdmin;

}

public static void setCheckAdmin(bool checkAdmin)

{

BankClassBL.checkAdmin = checkAdmin;

}

public static bool getLoginCheck()

{

return loginCheck;

}

public static void setLoginCheck(bool loginCheck)

{

BankClassBL.loginCheck = loginCheck;

}

public static bool getCheckUser()

{

return checkUser;

}

public static void setCheckUser(bool checkUser)

{

BankClassBL.checkUser = checkUser;

}

public static bool getMainMenuLogout()

{

return logout;

}

public static void setMainMenuLogout(bool logout)

{

BankClassBL.logout = logout;

}

public static string getCurrentRole()

{

return currentRole;

}

public static void setCurrentRole(string currentRole)

{

BankClassBL.currentRole = currentRole;

}

public static LoanClassBL returnProvinceAsProvinceObject(int option, string name)

{

DateTime dateTime = DateTime.Now;

if (option == 1)

{

LoanSindh loanProvince = new LoanSindh(dateTime, "Sindh", 0, 0, 0, "N/A",name);

return loanProvince;

}

else if (option == 2)

{

LoanKPK loanProvince = new LoanKPK(dateTime, "KPK", 0, 0, 0, "N/A", name);

return loanProvince;

}

else if (option == 3)

{

LoanBalochistan loanProvince = new LoanBalochistan(dateTime, "Balochistan", 0, 0, 0, "N/A", name);

return loanProvince;

}

else if (option == 4)

{

LoanPunjab loanProvince = new LoanPunjab(dateTime, "Punjab", 0, 0, 0, "N/A", name);

return loanProvince;

}

return null;

}

}

**PersonClassBL**

internal class PersonClassBL

{

public PersonClassBL(string userName, string password, string role)

{

this.userName = userName;

this.password = password;

this.role = role;

}

public PersonClassBL()

{

}

internal PersonClassDL Aggrigation\_1\_to\_Many

{

get => default;

set

{

}

}

protected string userName;

protected string password;

protected string role;

public string UserName { get => userName; set => userName = value; }

public string Password { get => password; set => password = value; }

public string Role { get => role; set => role = value; }

public string getName()

{

return userName;

}

public void setName(string name)

{

this.userName = name;

}

public string getPassword()

{

return password;

}

public void setPassword(string pass)

{

this.password = pass;

}

public string getRole()

{

return role;

}

public void setRole(string role)

{

this.role = role;

}

public virtual string getCreditCardNo()

{

return null;

}

public virtual void setCreditCardNo(string creditCardNo)

{

}

public virtual FundsBL getFund()

{

return null;

}

public virtual void setFund(FundsBL fund)

{

}

public virtual string getClientReviews()

{

return null;

}

public virtual void setClientReviews(string clientReviews)

{

}

public virtual LoanClassBL getLoans()

{

return null;

}

public virtual void setLoans(LoanClassBL loans)

{

}

public virtual string getComplaint()

{

return null;

}

public virtual void setComplaint(string complaints)

{

}

public virtual float getCash()

{

return 0;

}

public virtual void setCash(float cash)

{

}

public virtual int getDepositions()

{

return 0;

}

public virtual void setDepositions(int deposition)

{

}

public virtual int getTransactions()

{

return 0;

}

public virtual void setTransactions(int transactions)

{

}

public virtual bool getCheckBook()

{

return false;

}

public virtual void setCheckBook(bool checkBook)

{

}

public virtual bool getNewCard()

{

return false;

}

public virtual void setNewCard(bool newCard)

{

}

public virtual bool getReviewCheck()

{

return false;

}

public virtual void setReviewCheck(bool reviewCheck)

{

}

}

**AdminClassBL**

internal class AdminClassBL:PersonClassBL

{

public AdminClassBL(string name, string pass, string role) : base(name, pass, role)

{

}

}

**ClientClassBL**

internal class ClientClassBL : PersonClassBL

{

public ClientClassBL()

{

}

public ClientClassBL(string userName, string password, string role, string creditCardNo, string clientReviews, float cash, int depositions, int transactions, bool checkBook, bool newCard, bool reviewCheck, LoanClassBL loanObj) : base(userName, password, role)

{

this.creditCardNo = creditCardNo;

this.clientReviews = clientReviews;

this.cash = cash;

this.depositions = depositions;

this.transactions = transactions;

this.checkBook = checkBook;

this.newCard = newCard;

this.reviewCheck = reviewCheck;

this.loans = loanObj;

this.fund = new FundsBL();

complaints = "N/A";

}

private string creditCardNo;

private string clientReviews;

private float cash;

private int depositions;

private int transactions;

private bool checkBook;

private bool newCard;

private bool reviewCheck;

private FundsBL fund;

private string complaints;

private LoanClassBL loans;

public string CreditCardNo { get => creditCardNo; set => creditCardNo = value; }

public string ClientReviews { get => clientReviews; set => clientReviews = value; }

public float Cash { get => cash; set => cash = value; }

public int Depositions { get => depositions; set => depositions = value; }

public int Transactions { get => transactions; set => transactions = value; }

public bool CheckBook { get => checkBook; set => checkBook = value; }

public bool NewCard { get => newCard; set => newCard = value; }

public bool ReviewCheck { get => reviewCheck; set => reviewCheck = value; }

internal FundsBL Fund { get => fund; set => fund = value; }

public string Complaints { get => complaints; set => complaints = value; }

internal LoanClassBL Loans { get => loans; set => loans = value; }

public override string getCreditCardNo()

{

return creditCardNo;

}

public override void setCreditCardNo(string creditCardNo)

{

this.creditCardNo = creditCardNo;

}

public override FundsBL getFund()

{

return fund;

}

public override void setFund(FundsBL fund)

{

this.fund = fund;

}

public override string getClientReviews()

{

return clientReviews;

}

public override void setClientReviews(string clientReviews)

{

this.clientReviews = clientReviews;

}

public override LoanClassBL getLoans()

{

return loans;

}

public override void setLoans(LoanClassBL loans)

{

this.loans = loans;

}

public override float getCash()

{

return cash;

}

public override void setCash(float cash)

{

this.cash = cash;

}

public override int getDepositions()

{

return depositions;

}

public override void setDepositions(int deposition)

{

this.depositions = deposition;

}

public override int getTransactions()

{

return transactions;

}

public override void setTransactions(int transactions)

{

this.transactions = transactions;

}

public override bool getCheckBook()

{

return checkBook;

}

public override void setCheckBook(bool checkBook)

{

this.checkBook = checkBook;

}

public override bool getNewCard()

{

return newCard;

}

public override void setNewCard(bool newCard)

{

this.newCard = newCard;

}

public override bool getReviewCheck()

{

return reviewCheck;

}

public override void setReviewCheck(bool reviewCheck)

{

this.reviewCheck = reviewCheck;

}

public override string getComplaint()

{

return complaints;

}

public override void setComplaint(string complaint)

{

this.complaints = complaint;

}

}

**FundsBL**

internal class FundsBL

{

public FundsBL(bool funds, int fundPayed)

{

this.fundPayed = fundPayed;

this.funds = funds;

}

public FundsBL()

{

}

internal ClientClassBL Aggrigation\_1\_to\_1

{

get => default;

set

{

}

}

private bool funds;

private int fundPayed;

public bool Funds { get => funds; set => funds = value; }

public int FundPayed { get => fundPayed; set => fundPayed = value; }

public int getFundPayed()

{

return fundPayed;

}

public void setFundPayed(int fundPayed)

{

this.fundPayed = fundPayed;

}

public bool getFunds()

{

return funds;

}

public void setFunds(bool funds)

{

this.funds = funds;

}

}

class CopyOfFundsBL

{

public CopyOfFundsBL(bool funds, int fundPayed)

{

this.fundPayed = fundPayed;

this.funds = funds;

}

public CopyOfFundsBL()

{

}

internal ClientClassBL Aggrigation\_1\_to\_1

{

get => default;

set

{

}

}

private bool funds;

private int fundPayed;

public int getFundPayed()

{

return fundPayed;

}

public void setFundPayed(int fundPayed)

{

this.fundPayed = fundPayed;

}

public bool getFunds()

{

return funds;

}

public void setFunds(bool funds)

{

this.funds = funds;

}

}

**LoanClassBL**

internal class LoanClassBL

{

private string name;

private float loan;

private float loanRequested;

private string loanStatus;

private DateTime dateTime;

private string province;

private float interest;

public LoanClassBL(DateTime dateTime, string province, float loanRequested, float loan, float interest, string loanStatus,string name)

{

this.dateTime = dateTime;

this.interest = interest;

this.province = province;

this.loan = loan;

this.loanRequested = loanRequested;

this.loanStatus = loanStatus;

this.name = name;

}

public LoanClassBL(float loan, float interest)

{

this.loan = loan;

this.interest = interest;

}

public LoanClassBL()

{

}

internal ClientClassBL Aggrigation\_1to1

{

get => default;

set

{

}

}

public string Name { get => name; set => name = value; }

protected float Loan { get => loan; set => loan = value; }

protected float LoanRequested { get => loanRequested; set => loanRequested = value; }

protected string LoanStatus { get => loanStatus; set => loanStatus = value; }

protected DateTime DateTime { get => dateTime; set => dateTime = value; }

protected string Province { get => province; set => province = value; }

protected float Interest { get => interest; set => interest = value; }

public void setLoan(float loan)

{

this.loan = loan;

}

public float getLoan()

{

return loan;

}

public void setLoanStatus(string status)

{

this.loanStatus = status;

}

public string getLoanStatus()

{

return loanStatus;

}

public void setLoanRequested(float loanRequested)

{

this.loanRequested = loanRequested;

}

public float getLoanRequested()

{

return loanRequested;

}

public void setDateTime(DateTime dateTime)

{

this.dateTime = dateTime;

}

public DateTime getDateTime()

{

return dateTime;

}

public void setProvince(string province)

{

this.province = province;

}

public string getProvince()

{

return province;

}

public void setName(string name)

{

this.name = name;

}

public string getName()

{

return name;

}

public void setInterest(float interest)

{

this.interest = interest;

}

public float getInterest()

{

return interest;

}

public virtual float calculateInterest()

{

return 0;

}

public virtual float calculateTotalLoan()

{

return 0;

}

public virtual float calculateInterest(float loan)

{

return 0;

}

}

**LoanBalochistan**

internal class LoanBalochistan : LoanClassBL

{

public LoanBalochistan()

{

}

public LoanBalochistan(DateTime dateTime, string province, float loanTaken, float loan, float interest, string loanStatus,string name) : base(dateTime, province, loanTaken, loan, interest, loanStatus,name)

{

}

public override float calculateInterest()

{

float loan = base.getLoanRequested();

float interest = 0.1F \* loan;

return interest;

}

public override float calculateInterest(float loan)

{

float interest = loan \* 0.1F;

return interest;

}

public override float calculateTotalLoan()

{

float loan = base.getLoanRequested();

float loanToPay = loan + calculateInterest();

return loanToPay;

}

}

**LoanKPK**

internal class LoanKPK : LoanClassBL

{

public LoanKPK()

{

}

public LoanKPK(DateTime dateTime, string province, float loanTaken, float loan, float interest, string loanStatus,string name) : base(dateTime, province, loanTaken, loan, interest, loanStatus, name)

{

}

public override float calculateInterest()

{

float loan = base.getLoanRequested();

float interest = 0.15F \* loan;

return interest;

}

public override float calculateTotalLoan()

{

float loan = base.getLoanRequested();

float loanToPay = loan + calculateInterest();

return loanToPay;

}

}

**LoanPunjab**

internal class LoanPunjab : LoanClassBL

{

public LoanPunjab()

{

}

public LoanPunjab(DateTime dateTime, string province, float loanTaken, float loan, float interest, string loanStatus, string name) : base(dateTime, province, loanTaken, loan, interest, loanStatus, name)

{

}

public override float calculateInterest()

{

float loan = base.getLoanRequested();

float interest = 0.07F \* loan;

return interest;

}

public override float calculateTotalLoan()

{

float loan = base.getLoanRequested();

float loanToPay = loan + calculateInterest();

return loanToPay;

}

}

**LoanSindh**

internal class LoanSindh : LoanClassBL

{

public LoanSindh()

{

}

public LoanSindh(DateTime dateTime, string province, float loanTaken, float loan, float interest, string loanStatus, string name) : base(dateTime, province, loanTaken, loan, interest, loanStatus, name)

{

}

public override float calculateInterest()

{

float loan = base.getLoanRequested();

float interest = 0.05F \* loan;

return interest;

}

public override float calculateTotalLoan()

{

float loan = base.getLoanRequested();

float loanToPay = loan + calculateInterest();

return loanToPay;

}

}

**Forms Codes**

**MainMenuForm**

public partial class MainMenu : Form

{

public MainMenu()

{

InitializeComponent();

bankCodeVisible(false);

invisibleAll();

signInPnl.Visible = false;

}

private void guna2Button1\_Click(object sender, EventArgs e)

{

if (bankCodeValidation())

{

BankClassBL.setCurrentRole("Admin");

bankCodeVisible(false);

invisibleAll();

signInPnl.Visible = true;

}

}

private void bankCodeVisible(bool set)

{

bankCodeTxtBx.Visible = set;

bankCodeBtn.Visible = set;

}

private bool bankCodeValidation()

{

bool check = PersonClassDL.checkAdminBankCode(bankCodeTxtBx.Text);

if (string.IsNullOrEmpty(bankCodeTxtBx.Text.Trim()))

{

errorProvider.SetError(bankCodeTxtBx, "Enter Bank Code");

return false;

}

else if (!check)

{

errorProvider.SetError(bankCodeTxtBx, "Incorrect Bank Code");

return false;

}

else if (check)

{

return true;

}

return false;

}

private void bankCodeTxtBx\_TextChanged(object sender, EventArgs e)

{

}

private void guna2Button1\_Click\_1(object sender, EventArgs e)

{

PersonClassDL.storeData();

this.Close();

}

private void enterAdminBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

emptyAll();

if (bankCodeTxtBx.Visible == false)

{

bankCodeVisible(true);

}

}

private void enterClientBtn\_Click(object sender, EventArgs e)

{

bankCodeVisible(false);

BankClassBL.setCurrentRole("Client");

invisibleAll();

emptyAll();

signInPnl.Visible = true;

}

private void adminSignUpPnl1\_Load(object sender, EventArgs e)

{

}

private void signInBtn\_Click(object sender, EventArgs e)

{

int check = PersonClassDL.checkUser(SIuserNameTxtBx.Text, SIpasswordTxtBx.Text, BankClassBL.getCurrentRole());

if (signInValidation())

{

this.Hide();

BankClassBL.setCurrentUserIndex(check);

if (BankClassBL.getCurrentRole() == "Admin")

{

AdminMenuForm a = new AdminMenuForm();

a.Show();

}

else if (BankClassBL.getCurrentRole() == "Client")

{

ClientMenuForm c = new ClientMenuForm();

c.Show();

}

}

else

{

MessageBox.Show("Invalid User");

}

}

bool signInValidation()

{

int check = PersonClassDL.checkUser(SIuserNameTxtBx.Text, SIpasswordTxtBx.Text, BankClassBL.getCurrentRole());

if (string.IsNullOrEmpty(SIpasswordTxtBx.Text.Trim()))

{

errorProvider.SetError(SIpasswordTxtBx, "Enter Password");

return false;

}

if (string.IsNullOrEmpty(SIuserNameTxtBx.Text.Trim()))

{

errorProvider.SetError(SIuserNameTxtBx, "Enter Name");

return false;

}

if (check == -1)

{

errorProvider.SetError(SIpasswordTxtBx, "Incorrect password");

errorProvider.SetError(SIuserNameTxtBx, "Incorrect Name");

return false;

}

else if (check != -1)

{

return true;

}

return false;

}

private void createAccountBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

emptyAll();

if (BankClassBL.getCurrentRole() == "Admin")

{

adminSUPnl.Visible = true;

}

if (BankClassBL.getCurrentRole() == "Client")

{

clientSUPnl.Visible = true;

}

}

private void CSignUpBtn\_Click(object sender, EventArgs e)

{

if(clientSuValidation())

{

LoanClassBL loanObj = BankClassBL.returnProvinceAsProvinceObject(domecileCmBx.SelectedIndex + 1, CSUuserNameTxtBx.Text);

int check = PersonClassDL.checkUser(CSUuserNameTxtBx.Text, CSUpasswordTxtBx.Text, BankClassBL.getCurrentRole());

if (check != -1)

{

MessageBox.Show("User Already Exists, Try Different Name And Password");

}

else

{

ClientClassBL user = new ClientClassBL(CSUuserNameTxtBx.Text, CSUpasswordTxtBx.Text, "Client", credCardNoTxtBx.Text, "N/A", 0, 0, 0, false, false, false, loanObj);

PersonClassDL.addClientToList(user);

PersonClassDL.storeData();

MessageBox.Show("User is Added");

}

invisibleAll();

emptyAll();

signInPnl.Visible = true;

}

}

bool clientSuValidation()

{

if (string.IsNullOrEmpty(CSUpasswordTxtBx.Text.Trim()) || CSUpasswordTxtBx.Text.Length < 8)

{

errorProvider.SetError(CSUpasswordTxtBx, "Enter Password 8 Chars Min");

return false;

}

if (string.IsNullOrEmpty(CSUuserNameTxtBx.Text.Trim()) || CSUuserNameTxtBx.Text.Length < 3)

{

errorProvider.SetError(CSUuserNameTxtBx, "Enter Name 3 chars min");

return false;

}

if (string.IsNullOrEmpty(credCardNoTxtBx.Text.Trim()))

{

errorProvider.SetError(credCardNoTxtBx, "Enter CreditCardNo");

return false;

}

if (string.IsNullOrEmpty(domecileCmBx.Text.Trim()))

{

errorProvider.SetError(domecileCmBx, "Select a Place of Domecile");

return false;

}

return true;

}

private void CSUsignInInsteadBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

emptyAll();

signInPnl.Visible = true;

}

private void adminSignUpBtn\_Click(object sender, EventArgs e)

{

if(adminSUvalidation())

{

int check = PersonClassDL.checkUser(ASUuserNameTxtBx.Text, ASUpasswordTxtBx.Text, BankClassBL.getCurrentRole());

if (check != -1)

{

MessageBox.Show("User Already Exists, Try Different Name And Password");

}

else

{

AdminClassBL user = new AdminClassBL(ASUuserNameTxtBx.Text, ASUpasswordTxtBx.Text, BankClassBL.getCurrentRole());

PersonClassDL.addAdminToList(user);

PersonClassDL.storeData();

MessageBox.Show("User is Added");

}

invisibleAll();

emptyAll();

signInPnl.Visible = true;

}

}

bool adminSUvalidation()

{

if (string.IsNullOrEmpty(ASUpasswordTxtBx.Text.Trim()) || ASUpasswordTxtBx.Text.Length < 8)

{

errorProvider.SetError(ASUpasswordTxtBx, "Enter Password");

return false;

}

if (string.IsNullOrEmpty(ASUuserNameTxtBx.Text.Trim())|| ASUuserNameTxtBx.Text.Length <3)

{

errorProvider.SetError(ASUuserNameTxtBx, "Enter Password");

return false;

}

return true;

}

private void ASUsignInInstead\_Click(object sender, EventArgs e)

{

invisibleAll();

emptyAll();

signInPnl.Visible = true;

}

void invisibleAll()

{

clientSUPnl.Visible = false;

signInPnl.Visible = false;

adminSUPnl.Visible = false;

}

private void adminSUPnl\_Paint(object sender, PaintEventArgs e)

{

}

void emptyAll()

{

CSUpasswordTxtBx.Text = string.Empty;

CSUuserNameTxtBx.Text = string.Empty;

credCardNoTxtBx.Text = string.Empty;

ASUpasswordTxtBx.Text = string.Empty;

ASUuserNameTxtBx.Text = string.Empty;

SIpasswordTxtBx.Text = string.Empty;

SIuserNameTxtBx.Text = string.Empty;

}

private void CSUuserNameTxtBx\_TextChanged(object sender, EventArgs e)

{

}

private void CSUpasswordTxtBx\_TextChanged(object sender, EventArgs e)

{

}

private void credCardNoTxtBx\_TextChanged(object sender, EventArgs e)

{

}

private void domecileCmBx\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void credCardNoTxtBx\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b') // Allow backspace (\b) for editing

{

e.Handled = true; // Ignore the input character

}

}

private void MainMenu\_Load(object sender, EventArgs e)

{

}

}

**AdminMenuForm**

public partial class AdminMenuForm : Form

{

public AdminMenuForm()

{

InitializeComponent();

invisibleAll();

}

private void feedBackBtn\_Click(object sender, EventArgs e)

{

}

private void showClintsBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

showClientsPnl1.showData();

showClientsPnl1.Visible = true;

}

private void loanRequestsBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

showLoanRequestsPnl1.showData();

showLoanRequestsPnl1.Visible = true;

}

private void showComplaintsBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

showComplaintsPnl1.showData();

showComplaintsPnl1.Visible = true;

}

private void transHistoryBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

transHistoryPnl1.showData();

transHistoryPnl1.Visible=true;

}

private void showDonationsBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

depositionHistoryPnl1.showData();

depositionHistoryPnl1.Visible=true;

}

private void bankBalanceBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

showBankBalancePnl1.showData();

showBankBalancePnl1.Visible = true;

}

private void searchClientBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

searchClientPnl1.Visible = true;

searchClientPnl1.setvisible(false);

}

private void removeClientBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

removeClientPnl1.Visible = true;

}

private void showReviewBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

showFeedBackPnl1.showData();

showFeedBackPnl1.Visible=true;

}

private void editAccountBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

adminEditPnl1.Visible = true;

}

private void guna2Button1\_Click(object sender, EventArgs e)

{

this.Dispose();

MainMenu m = new MainMenu();

m.Show();

}

private void guna2CirclePictureBox1\_Click(object sender, EventArgs e)

{

this.Dispose();

MainMenu m = new MainMenu();

m.Show();

}

void invisibleAll()

{

showClientsPnl1.Visible = false;

showLoanRequestsPnl1.Visible = false;

showComplaintsPnl1.Visible = false;

transHistoryPnl1.Visible = false;

depositionHistoryPnl1.Visible = false;

showBankBalancePnl1.Visible = false;

searchClientPnl1.Visible = false;

removeClientPnl1.Visible = false;

showFeedBackPnl1.Visible = false;

adminEditPnl1.Visible = false;

}

}

**EditAdminPnl**

public partial class AdminEditPnl : UserControl

{

public AdminEditPnl()

{

InitializeComponent();

}

private void confirmBtn\_Click(object sender, EventArgs e)

{

editAccount();

}

public void editAccount()

{

PersonClassBL users = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

if(validate())

{

if (passwordTxtBx.Text.Length < 8 || userNameTxtBx.Text.Length < 3 || bankCodeTxtBx.Text.Length < 8)

{

showInvalidPass();

}

else

{

users.setName(userNameTxtBx.Text);

users.setPassword(passwordTxtBx.Text);

PersonClassDL.setBankCode(bankCodeTxtBx.Text);

showAccountEditMsg();

PersonClassDL.storeData();

}

}

emptyAll();

}

bool validate()

{

if (string.IsNullOrEmpty(userNameTxtBx.Text.Trim()))

{

errorProvider1.SetError(userNameTxtBx, "Enter New Name");

return false;

}

if (string.IsNullOrEmpty(passwordTxtBx.Text.Trim()))

{

errorProvider1.SetError(passwordTxtBx, "Enter New Password");

return false;

}

if (string.IsNullOrEmpty(bankCodeTxtBx.Text.Trim()))

{

errorProvider1.SetError(bankCodeTxtBx, "Enter New BankCode");

return false;

}

return true;

}

public void showInvalidPass()

{

MessageBox.Show("Password & BankCode should be atleast 8 characters long\n & Name Should Be Atleast 3 Characters Long");

}

public void showAccountEditMsg()

{

MessageBox.Show("Your Account Has Been Edited");

}

void emptyAll()

{

passwordTxtBx.Text = string.Empty;

userNameTxtBx.Text = string.Empty;

}

}

**Diposition HistoryPnl**

public DepositionHistoryPnl()

{

InitializeComponent();

}

public void showData()

{

List<ClientClassBL> clients = PersonClassDL.getClientList();

DataGrid.DataSource = clients;

invisibleEverything();

DataGrid.Columns["UserName"].Visible = true;

DataGrid.Columns["Password"].Visible = true;

DataGrid.Columns["Cash"].Visible = true;

DataGrid.Columns["Depositions"].Visible = true;

}

void invisibleEverything()

{

DataGrid.Columns["UserName"].Visible = false;

DataGrid.Columns["Password"].Visible = false;

DataGrid.Columns["Role"].Visible = false;

DataGrid.Columns["CreditCardNo"].Visible = false;

DataGrid.Columns["ClientReviews"].Visible = false;

DataGrid.Columns["Cash"].Visible = false;

DataGrid.Columns["Depositions"].Visible = false;

DataGrid.Columns["Transactions"].Visible = false;

DataGrid.Columns["CheckBook"].Visible = false;

DataGrid.Columns["NewCard"].Visible = false;

DataGrid.Columns["ReviewCheck"].Visible = false;

DataGrid.Columns["Complaints"].Visible = false;

}

}

**RemoveClientPnl**

public partial class RemoveClientPnl : UserControl

{

public RemoveClientPnl()

{

InitializeComponent();

}

private void removeBtn\_Click(object sender, EventArgs e)

{

if (validate())

{

int check = PersonClassDL.checkUser(searchNameTxtBx.Text, searchPassTxtBx.Text, "Client");

PersonClassDL.delUserAccount(check);

MessageBox.Show("User Account has Been Removed ");

}

emptyAll();

}

bool validate()

{

int check = PersonClassDL.checkUser(searchNameTxtBx.Text, searchPassTxtBx.Text, "Client");

if (string.IsNullOrEmpty(searchPassTxtBx.Text.Trim()))

{

errorProvider1.SetError(searchPassTxtBx, "Enter Password");

return false;

}

if (string.IsNullOrEmpty(searchNameTxtBx.Text.Trim()))

{

errorProvider1.SetError(searchNameTxtBx, "Enter Name");

return false;

}

if (check == -1)

{

errorProvider1.SetError(searchPassTxtBx, "Incorrect password");

errorProvider1.SetError(searchNameTxtBx, "Incorrect Name");

return false;

}

else if (check != -1)

{

return true;

}

emptyAll();

return false;

}

void emptyAll()

{

searchNameTxtBx.Text = string.Empty;

searchPassTxtBx.Text = string.Empty;

}

}

**SearchClientPnl**

public partial class SearchClientPnl : UserControl

{

public SearchClientPnl()

{

InitializeComponent();

detailsPnl.Visible = false;

}

public void setvisible(bool set)

{

detailsPnl.Visible = set;

}

private void sendBtn\_Click(object sender, EventArgs e)

{

if (validate())

{

int check = PersonClassDL.checkUser(searchNameTxtBx.Text, searchPassTxtBx.Text, "Client");

PersonClassBL user = PersonClassDL.takeUserFromList(check);

LoanClassBL loan = user.getLoans();

FundsBL fund = user.getFund();

guna2HtmlLabel12.Text = "Name is: " + user.getName();

guna2HtmlLabel1.Text = "Password is: " + user.getPassword();

guna2HtmlLabel2.Text = "Bank Balance is: " + user.getCash();

guna2HtmlLabel3.Text = "loan to pay is:: " + loan.getLoan();

guna2HtmlLabel4.Text = "Interest is: " + loan.getInterest();

guna2HtmlLabel5.Text = " loan Requested was: " + loan.getLoanRequested();

guna2HtmlLabel6.Text = "Total Donation Given is: " + fund.getFundPayed();

guna2HtmlLabel7.Text = "Complaint is: " + user.getComplaint();

guna2HtmlLabel8.Text = "Review is: " + user.getClientReviews();

detailsPnl.Visible = true;

}

emptyAll();

}

void emptyAll()

{

searchNameTxtBx.Text = string.Empty;

searchPassTxtBx.Text = string.Empty;

}

bool validate()

{

int check = PersonClassDL.checkUser(searchNameTxtBx.Text, searchPassTxtBx.Text, "Client");

if (string.IsNullOrEmpty(searchPassTxtBx.Text.Trim()))

{

errorProvider1.SetError(searchPassTxtBx, "Enter Password");

return false;

}

if (string.IsNullOrEmpty(searchNameTxtBx.Text.Trim()))

{

errorProvider1.SetError(searchNameTxtBx, "Enter Name");

return false;

}

if (check == -1)

{

errorProvider1.SetError(searchPassTxtBx, "Incorrect password");

errorProvider1.SetError(searchNameTxtBx, "Incorrect Name");

return false;

}

else if (check != -1)

{

return true;

}

emptyAll();

return false;

}

}

**showBankBalancePnl**

public partial class showBankBalancePnl : UserControl

{

public showBankBalancePnl()

{

InitializeComponent();

}

public void showData()

{

showBalanceLbl.Text = " " + PersonClassDL.getBankBalance(); ;

}

private void showBankBalancePnl\_Load(object sender, EventArgs e)

{

}

}

**showClientsPnl** public partial class showClientsPnl : UserControl

{

public showClientsPnl()

{

InitializeComponent();

}

public void showData()

{

List <ClientClassBL> clients = PersonClassDL.getClientList();

DataGrid.DataSource = clients;

invisibleEverything();

DataGrid.Columns["UserName"].Visible = true;

DataGrid.Columns["Password"].Visible = true;

DataGrid.Columns["Cash"].Visible = true;

DataGrid.Columns["CreditCardNo"].Visible = true;

PersonClassDL.storeData();

}

private void showClientsPnl\_Load(object sender, EventArgs e)

{

}

private void DataGrid\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

**showComplaintsPnl**

public partial class showComplaintsPnl : UserControl

{

public showComplaintsPnl()

{

InitializeComponent();

}

public void showData()

{

List<ClientClassBL> clients = PersonClassDL.getClientList();

DataGrid.DataSource = clients;

invisibleEverything();

DataGrid.Columns["UserName"].Visible = true;

DataGrid.Columns["Complaints"].Visible = true;

}

}

**showFeedbackPnl**

public partial class showFeedBackPnl : UserControl

{

public showFeedBackPnl()

{

InitializeComponent();

}

public void showData()

{

List<ClientClassBL> clients = PersonClassDL.getClientList();

List<ClientClassBL> copyclients = new List<ClientClassBL>();

foreach (ClientClassBL client in clients)

{

if (client.getReviewCheck() != true)

{

copyclients.Add(client);

}

}

DataGrid.DataSource = copyclients; ;

invisibleEverything();

DataGrid.Columns["UserName"].Visible = true;

DataGrid.Columns["ClientReviews"].Visible = true;

}

}

**showLoanRequestPnl**

public ShowLoanRequestsPnl()

{

InitializeComponent();

}

public void showData()

{

List<ClientClassBL> clients = PersonClassDL.getClientList();

List<LoanClassBL> loaners = new List<LoanClassBL>();

foreach (ClientClassBL client in clients)

{

if (client.getLoans().getLoanStatus() == "Requested")

{

loaners.Add(client.getLoans());

}

}

DataGrid.DataSource = loaners;

// invisibleEverything();

DataGrid.Columns["Name"].Visible = true;

PersonClassDL.storeData();

}

void invisibleEverything()

{

DataGrid.Columns["Loan"].Visible = false;

DataGrid.Columns["LoanRequested"].Visible = false;

DataGrid.Columns["LoanStatus"].Visible = false;

DataGrid.Columns["DateTime"].Visible = false;

DataGrid.Columns["Province"].Visible = false;

DataGrid.Columns["Interest"].Visible = false;

DataGrid.Columns["Name"].Visible = false;

}

**TransHistoryPnl**

public void showData()

{

List<ClientClassBL> clients = PersonClassDL.getClientList();

DataGrid.DataSource = clients;

invisibleEverything();

DataGrid.Columns["UserName"].Visible = true;

DataGrid.Columns["Password"].Visible = true;

DataGrid.Columns["Cash"].Visible = true;

DataGrid.Columns["Transactions"].Visible = true;

}

**ClientPanels**

**ClientMenuFormPnl**

public partial class ClientMenuForm : Form

{

public ClientMenuForm()

{

InitializeComponent();

invisibleAll();

}

private void showBalanceBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

checkBalancePnl1.Visible = true;

checkBalancePnl1.showBalance();

}

private void guna2HtmlLabel1\_Click(object sender, EventArgs e)

{

}

private void withDrawBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

withdrawPnl1.Visible = true;

withdrawPnl1.showBalance();

}

private void depositCashBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

depositPnl1.Visible = true;

depositPnl1.showBalance();

}

private void requestLoanBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

loanRequestPnl1.Visible=true;

loanRequestPnl1.loanStatus();

}

private void payLoanBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

loanPayPnl1.Visible = true;

loanPayPnl1.showBalance();

}

private void sendCashBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

sendCashPnl1.Visible = true;

sendCashPnl1.showBalance();

}

private void guna2Button3\_Click(object sender, EventArgs e)

{

invisibleAll();

complaintPnl1.Visible = true;

}

private void donateBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

donationPnl1.Visible = true;

donationPnl1.showBalance();

}

private void feedBackBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

feedBackPnl1.Visible = true;

feedBackPnl1.setTxtBxInvisible();

}

private void editAccountBtn\_Click(object sender, EventArgs e)

{

invisibleAll();

clientEditPnl1.Visible = true;

}

private void guna2CirclePictureBox1\_Click(object sender, EventArgs e)

{

this.Dispose();

MainMenu m = new MainMenu();

m.Show();

}

void invisibleAll()

{

checkBalancePnl1.Visible = false;

withdrawPnl1.Visible = false;

depositPnl1.Visible = false;

loanRequestPnl1.Visible = false;

loanPayPnl1.Visible = false;

sendCashPnl1.Visible = false;

complaintPnl1.Visible = false;

donationPnl1.Visible = false;

feedBackPnl1.Visible = false;

clientEditPnl1.Visible = false;

}

private void guna2Button1\_Click(object sender, EventArgs e)

{

this.Dispose();

MainMenu m = new MainMenu();

m.Show();

}

}

**checkBalancePnl**

public partial class CheckBalancePnl : UserControl

{

public CheckBalancePnl()

{

InitializeComponent();

}

private void CheckBalancePnl\_Load(object sender, EventArgs e)

{

}

public void showBalance()

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

LoanClassBL loan = user.getLoans();

showBalanceLbl.Text = " " + user.getCash();

showLoanLbl.Text = " " + loan.getLoan();

}

private void guna2HtmlLabel1\_Click(object sender, EventArgs e)

{

}

private void showBalanceLbl\_Click(object sender, EventArgs e)

{

}

private void guna2HtmlLabel3\_Click(object sender, EventArgs e)

{

}

}

**ClientEditPnl**

public partial class ClientEditPnl : UserControl

{

public ClientEditPnl()

{

InitializeComponent();

}

private void feedBackBtn\_Click(object sender, EventArgs e)

{

editAccount();

}

public void editAccount()

{

PersonClassBL users = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

if (passwordTxtBx.Text.Length < 8 || userNameTxtBx.Text.Length < 3)

{

showInvalidPass();

}

else

{

users.setName(userNameTxtBx.Text);

users.setPassword(passwordTxtBx.Text);

showAccountEditMsg();

PersonClassDL.storeData();

}

emptyAll();

}

public void showInvalidPass()

{

MessageBox.Show("Password should b atleast 8 characters long\n & Name Should Be Atleast 3 Characters Long");

}

public void showAccountEditMsg()

{

MessageBox.Show("Your Account Has Been Edited");

}

void emptyAll()

{

passwordTxtBx.Text = string.Empty;

userNameTxtBx.Text = string.Empty;

}

private void ClientEditPnl\_Load(object sender, EventArgs e)

{

}

private void userNameTxtBx\_TextChanged(object sender, EventArgs e)

{

}

private void passwordTxtBx\_TextChanged(object sender, EventArgs e)

{

}

private void Header\_Click(object sender, EventArgs e)

{

}

}

**ComplaintPnl**

public partial class ComplaintPnl : UserControl

{

public ComplaintPnl()

{

InitializeComponent();

}

private void sendBtn\_Click(object sender, EventArgs e)

{

complaint();

}

public void complaint()

{

PersonClassBL users = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

string complaint = complaintTxtBx.Text;

PersonClassDL.storeData();

emptyAll();

}

void emptyAll()

{

complaintTxtBx.Text = string.Empty;

}

public void noMoreComplaints()

{

MessageBox.Show("You Cant Add More Than 3 Complaints");

}

}

**DepositPnl**

public partial class DepositPnl : UserControl

{

public DepositPnl()

{

InitializeComponent();

}

private void DepositPnl\_Load(object sender, EventArgs e)

{

}

public void showBalance()

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

showBalanceLbl.Text = " " + user.getCash();

}

private void withDrawBtn\_Click(object sender, EventArgs e)

{

cashDeposit();

}

public void cashDeposit()

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

string deposit = getDepositText();

string creditCard = getCreditCard();

if (creditCard == user.getCreditCardNo())

{

PersonClassDL.addBankBalance(int.Parse(deposit));

PersonClassDL.addUserCash(int.Parse(deposit));

PersonClassDL.addUserDepositions(int.Parse(deposit));

showDeposited(deposit);

setShowBalance(user.getCash());

PersonClassDL.storeData();

}

else

{

showWrongCreditCard();

}

emptyAll();

}

void emptyAll()

{

depositTxtBx.Text = string.Empty;

creditCardTxtBx.Text = string.Empty;

}

public void showDeposited(string deposit)

{

MessageBox.Show(deposit + "Rs have been deposited in Your Account");

}

public void showWrongCreditCard()

{

MessageBox.Show("Wrong Credit Card No");

}

public void showInvalid()

{

MessageBox.Show("Enter Digits only & CreditCard Number should be 16 characters Long");

}

public string getDepositText()

{

return depositTxtBx.Text;

}

public string getCreditCard()

{

return creditCardTxtBx.Text;

}

public void setShowBalance(float cash)

{

showBalanceLbl.Text = " " + cash;

}

private void creditCardTxtBx\_KeyPress(object sender, KeyPressEventArgs e)

{

}

private void depositTxtBx\_TextChanged(object sender, EventArgs e)

{

}

private void depositTxtBx\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b') // Allow backspace (\b) for editing

{

e.Handled = true; // Ignore the input character

}

}

}

**FeedBackPnl**

public partial class FeedBackPnl : UserControl

{

public FeedBackPnl()

{

InitializeComponent();

}

public void setTxtBxInvisible()

{

customTxtBx.Visible = false;

}

private void customChBx\_CheckedChanged(object sender, EventArgs e)

{

if(customChBx.Checked)

{

customTxtBx.Visible = true;

}

else if(!customChBx.Checked)

{

customTxtBx.Visible = false;

}

}

private void feedBackBtn\_Click(object sender, EventArgs e)

{

bool check = clientReview();

if (check == true)

{

MessageBox.Show("Thanks For Your FeedBack");

}

emptyAll();

}

public bool clientReview()

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

bool check = false;

if (excellentChBx.Checked == true)

{

check = true;

user.setReviewCheck(true);

user.setClientReviews("Excellent");

}

else if (veryBadChBx.Checked == true)

{

check = true;

user.setReviewCheck(true);

user.setClientReviews("Very Good");

}

else if (goodChBx.Checked == true)

{

check = true;

user.setReviewCheck(true);

user.setClientReviews("Good");

}

else if (averageChBx.Checked == true)

{

check = true;

user.setReviewCheck(true);

user.setClientReviews("Average");

}

else if (badChBx.Checked == true)

{

check = true;

user.setReviewCheck(true);

user.setClientReviews("Bad");

}

else if (veryBadChBx.Checked == true)

{

check = true;

user.setReviewCheck(true);

user.setClientReviews("Very Bad");

}

else if (worstChBx.Checked == true)

{

check = true;

user.setReviewCheck(true);

user.setClientReviews("Worst");

}

else if (customChBx.Checked == true)

{

string str = customTxtBx.Text;

if (str == "" || str == null)

{

check = false;

showEnterCustom();

}

else

{

check = true;

user.setReviewCheck(true);

user.setClientReviews(str);

}

}

else

{

check = false;

user.setReviewCheck(false);

enterValid();

}

return check;

}

public void showEnterCustom()

{

MessageBox.Show("Enter a Custom Review First");

}

public void enterValid()

{

MessageBox.Show("Check a Box");

}

void emptyAll()

{

excellentChBx.Checked = false;

veryGoodChBx.Checked = false;

goodChBx.Checked = false;

averageChBx.Checked = false;

badChBx.Checked= false;

veryBadChBx.Checked = false;

worstChBx.Checked = false;

customTxtBx.Visible = false;

customTxtBx.Text = string.Empty;

}

}

**DonationPnl**

public partial class DonationPnl : UserControl

{

public DonationPnl()

{

InitializeComponent();

}

public void showBalance()

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

showBalanceLbl.Text = " " + user.getCash();

}

private void donateBtn\_Click(object sender, EventArgs e)

{

clientFunds();

}

public void clientFunds()

{

PersonClassBL users = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

int fund = int.Parse(donationTxtBx.Text);

string creditCard = creditCardTxtBx.Text;

if (creditCard == users.getCreditCardNo() && users.getCash() > fund)

{

PersonClassDL.subtractUserCash(fund);

showBalanceLbl.Text = "" + users.getCash();

FundsBL funds = users.getFund();

funds.setFundPayed(funds.getFundPayed() + fund);

funds.setFunds(true);

PersonClassDL.storeData();

}

else if (creditCard != users.getCreditCardNo() || users.getCash() < fund)

{

showInvalidCreditCardNo();

}

}

public void showInvalidCreditCardNo()

{

MessageBox.Show("Your CreditCard Number is Wrong");

}

public void showDigitsOnly()

{

MessageBox.Show("Enter Digits Only");

}

private void creditCardTxtBx\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b') // Allow backspace (\b) for editing

{

e.Handled = true; // Ignore the input character

}

}

private void donationTxtBx\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b') // Allow backspace (\b) for editing

{

e.Handled = true; // Ignore the input character

}

}

}

**loanPayPnl**

public partial class LoanPayPnl : UserControl

{

public LoanPayPnl()

{

InitializeComponent();

}

public void showBalance()

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

showBalanceLbl.Text = " " + user.getCash();

LoanClassBL loan = user.getLoans();

showTotalLoanLbl.Text = "" + loan.getLoan();

}

private void withDrawBtn\_Click(object sender, EventArgs e)

{

loanPay();

}

public void loanPay()

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

string payLoan = "0";

LoanClassBL userLoan = user.getLoans();

if (userLoan.getLoan() == 0)

{

showNoLoans();

}

else if (userLoan.getLoan() > 0)

{

if (getCreditCard() == user.getCreditCardNo() && (user.getCash()) > int.Parse(payLoan))

{

if (float.Parse(payLoan) >= userLoan.getLoan())

{

PersonClassDL.addBankBalance(int.Parse(payLoan));

float payed;

payed = float.Parse(payLoan) - userLoan.getLoan();

user.setCash(user.getCash() + payed);

userLoan.setLoan(0);

showPayedMore(payed);

}

else if (float.Parse(payLoan) < userLoan.getLoan())

{

PersonClassDL.addBankBalance(int.Parse(payLoan));

userLoan.setLoan(userLoan.getLoan() - float.Parse(payLoan));

user.setCash(user.getCash() - float.Parse(payLoan));

showPayedLess(userLoan.getLoan());

}

}

else

{

showInvalidCreditCard();

}

}

emptyAll();

}

void emptyAll()

{

creditCardTxtBx.Text = string.Empty;

payLoanTxtBx.Text = string.Empty;

}

public void showNoLoans()

{

MessageBox.Show("You Dont Have Any Loans To Pay");

}

public void showInvalid()

{

MessageBox.Show("Enter Digits only & CreditCard Number Should be of 16 Characters only");

}

public string getPayLoan()

{

return payLoanTxtBx.Text;

}

public string getCreditCard()

{

return creditCardTxtBx.Text;

}

public void showPayedMore(float payed)

{

MessageBox.Show("You Have Payed More than you were owed\n Remaining money" + payed + " has been deposited to Your account");

}

public void showPayedLess(float payed)

{

MessageBox.Show("You Have Payed Less Money than you were owed\n Remaining Loan to pay is " + payed);

}

public void showInvalidCreditCard()

{

MessageBox.Show("Invalid Credit Card Number or Not Suficient Balance");

}

private void payLoanTxtBx\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b') // Allow backspace (\b) for editing

{

e.Handled = true; // Ignore the input character

}

}

private void creditCardTxtBx\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b') // Allow backspace (\b) for editing

{

e.Handled = true; // Ignore the input character

}

}

}

**LoanRequestPnl**

public partial class LoanRequestPnl : UserControl

{

public LoanRequestPnl()

{

InitializeComponent();

}

private void loanRequestBtn\_Click(object sender, EventArgs e)

{

loanRequest();

}

public void loanRequest()

{

if (string.IsNullOrEmpty(loanRequestTxtBx.Text.Trim()))

{

errorProvider1.SetError(loanRequestTxtBx, "Select a Place of Domecile");

}

else

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

string inputLoan = getLoanRequestText();

float loans = float.Parse(inputLoan);

LoanClassBL loan = user.getLoans();

if (loan.getLoan() > 0)

{

alreadyhaveLoan();

}

else if (loan.getLoanStatus() != "N/A")

{

showRequestSent();

}

else if (loan.getLoanStatus() == "N/A")

{

loan.setLoanRequested(loans);

loan.setLoanStatus("Requested");

}

PersonClassDL.storeData();

emptyAll();

}

}

void emptyAll()

{

loanRequestTxtBx.Text = string.Empty;

}

public void loanStatus()

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

LoanClassBL loan = user.getLoans();

if (loan.getLoanStatus() == "Approved")

{

float loans = loan.getLoanRequested();

float interest = loan.calculateInterest();

loan.setInterest(interest);

float totalLoan = interest + loans;

loan.setLoan(totalLoan);

setInterestLbl(interest);

setTotalLoan(totalLoan);

}

else if (loan.getLoanStatus() == "Denied")

{

requestDenied();

loan.setLoanStatus("N/A");

}

}

public void alreadyhaveLoan()

{

MessageBox.Show("You Already Have a Loan to Pay");

}

public void showRequestSent()

{

MessageBox.Show("You Have Already Sent a Request");

}

public string getLoanRequestText()

{

return loanRequestTxtBx.Text;

}

public void setInterestLbl(float interest)

{

showInterestLbl.Text = "" + interest;

}

public void setTotalLoan(float loan)

{

showTotalLoanLbl.Text = "" + loan;

}

public void requestDenied()

{

MessageBox.Show("Your Request for loan was Denied");

}

private void loanRequestTxtBx\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b') // Allow backspace (\b) for editing

{

e.Handled = true; // Ignore the input character

}

}

}

**SendCashPnl**

public partial class SendCashPnl : UserControl

{

public SendCashPnl()

{

InitializeComponent();

}

public void showBalance()

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

showBalanceLbl.Text = " " + user.getCash();

}

private void sendBtn\_Click(object sender, EventArgs e)

{

sendCash();

}

public void sendCash()

{

float sendMoney;

int reciever;

reciever = PersonClassDL.checkUser(recieverNameTxtBx.Text, reciverPassTxtBx.Text, "Client");

if (reciever == -1)

{

showNotSignedIn();

}

else if (reciever != -1)

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

PersonClassBL userR = PersonClassDL.takeUserFromList(reciever);

sendMoney = float.Parse(sendCashTxtBx.Text);

if ((user.getCash()) >= sendMoney)

{

user.setCash(user.getCash() - sendMoney);

userR.setCash(userR.getCash() + sendMoney);

showBalanceLbl.Text = "" + user.getCash();

PersonClassDL.storeData();

}

else if (user.getCash() < sendMoney)

{

showInsufficientBalance();

}

}

emptyAll();

}

void emptyAll()

{

recieverNameTxtBx.Text = string.Empty;

reciverPassTxtBx.Text= string.Empty;

sendCashTxtBx.Text = string.Empty;

}

public void showInsufficientBalance()

{

MessageBox.Show("You Donot Have Sufficient Balance");

}

public void showNotSignedIn()

{

MessageBox.Show("This User is Not Present");

}

public void showDigitsOnly()

{

MessageBox.Show("Enter Digits Only");

}

private void sendCashTxtBx\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b') // Allow backspace (\b) for editing

{

e.Handled = true; // Ignore the input character

}

}

}

**withdrawPnl**

public partial class WithdrawPnl : UserControl

{

public WithdrawPnl()

{

InitializeComponent();

}

private void withDrawTxtBx\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b') // Allow backspace (\b) for editing

{

e.Handled = true; // Ignore the input character

}

}

private void WithdrawPnl\_Load(object sender, EventArgs e)

{

}

public void showBalance()

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

showBalanceLbl.Text = " " + user.getCash();

}

private void withDrawBtn\_Click(object sender, EventArgs e)

{

cashWithdraw();

}

public void cashWithdraw()

{

PersonClassBL user = PersonClassDL.takeUserFromList(BankClassBL.getCurrentUserIndex());

string withdraw = getWithdrawText();

if (int.Parse(withdraw) <= (user.getCash()))

{

PersonClassDL.subtractUserCash(int.Parse(withdraw));

showWithdrawn(withdraw);

setBalanceLbl(user.getCash());

PersonClassDL.subtractBankBalance(int.Parse(withdraw));

PersonClassDL.addUserTransactions(int.Parse(withdraw));

PersonClassDL.storeData();

}

else if (int.Parse(withdraw) > (user.getCash()))

{

showInsufficientBalance();

}

emptyAll();

}

public void showMsg(string msg)

{

MessageBox.Show("Message " + msg);

}

public string getWithdrawText()

{

return withdrawTxtBx.Text;

}

public void showInsufficientBalance()

{

MessageBox.Show("You Dont Have Sufficient Ballance");

}

public void showWithdrawn(string withdraw)

{

MessageBox.Show("You Have Withdrawn " + withdraw + " Rs");

}

public void setBalanceLbl(float cash)

{

showBalanceLbl.Text = " " + cash;

}

void emptyAll()

{

withdrawTxtBx.Text = string.Empty;

}

private void withdrawTxtBx\_TextChanged(object sender, EventArgs e)

{

}

}

**Conclusion:**

The project achieves many of its goal but it is lacking in some of the places. I learned a lot of new things while writing this code which I probably wouldn’t be able to learn without practice. It was quite difficult and I face many challenges but I was able to face them. In the end I make this project with hard work. It was a great experience writing this code.