Blood Bank Management System

-Final Project-

Pankaja Isuranga Balasooriya

GAM/A-12212

DiSE – 40

ESOFT Metro Campus - Gampaha

Contents

[1.Introduction 2](#_Toc97326702)

[2.Waterfall method 3](#_Toc97326703)

[3.System Design 5](#_Toc97326704)

[4. Preview 8](#_Toc97326705)

[5. Coding 12](#_Toc97326706)

[6.Software Test 26](#_Toc97326707)

[7.Grant Chart 31](#_Toc97326708)

# 1.Introduction

The project titled Blood Bank Management system is a Blood Bank management software for stock management and doner management tool for a blood bank. This project is developed in Microsoft visual studio which mainly focuses on basic operations of a blood bank such as stock keeping, stock updating, doer registration, doner search, searching specific doners with specific blood group and updating doner data.

The blood bank management system is a windows application for 64-bit windows operating system which helps documentation process of a blood bank. This software is easy to use for beginners as well as advanced users. It features a familiar, attractive and efficient user interface with a strong searching and updating capabilities.

# 2.Waterfall method

Waterfall method for developing software is used for developing his project . it is very simple to understand and follow.

Requirement Definition

System and software design

Implementation and unit testing

Integration and system Testing

Operation and maintenance

Requirement Definition

The systems services, constrains, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

System and software design

The system design process partitions the requirements to either hardware or software systems. It establishes overall system architecture. Software design involves identifying and describing the fundamental software systems abstractions and their relationships.

Implementation and unit testing

During this stage, the software design is realized as a set of programmed or program units. Unit testing involves verifying hat each unit meets its specification.

Integration and system testing

The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been me. After testing the software system is delivered to the customer.

Operation and maintenance

The system is put into practical use. Maintenance involves correcting the errors which were not discovered in earlier stages of life cycle, improving he implementation of system units and enhancing he system units and enhancing he system’s services as a new requirements are discovered.

# 3.System Design

1. ER Diagram

Diagram

Description automatically generated

1. Use Case Diagram

Diagram

Description automatically generated

1. Class Diagram

Diagram

Description automatically generated

# 4. Preview

1. Login

Graphical user interface, application

Description automatically generated

1. Main Menu

Graphical user interface, application, PowerPoint

Description automatically generated

1. Doner Registration

Graphical user interface

Description automatically generated

1. Doner Details

Table

Description automatically generated

1. Graphical user interface, application

   Description automatically generatedSearch a doner through blood Group
2. Blood Recipients

Graphical user interface

Description automatically generated

1. Blood Bank Stock

Graphical user interface, application, table

Description automatically generated

1. Update Stock

Graphical user interface, application

Description automatically generated

# 5. Coding

1. Database Class

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MySql.Data.MySqlClient;

using System.Data;

namespace ProjectAlpha

{

class DatabaseClass

{

//create the instant variable

MySqlConnection mysqlConn;

MySqlCommand mysqlComm;

MySqlDataAdapter mysqlAdapt;

MySqlDataReader mysqlRead;

DataSet dsDoners;

//create memory variables

String qry;

//creating the connection Method

public MySqlConnection myCon()

{

//Create the memory variable

string srv, db, uid, pwd;

//Assigning the values

srv = "localhost"; db = "bloodbank"; uid = "root"; pwd = "";

//creating the statement

qry = "SERVER=" + srv + ";" + "DATABASE=" + db + ";" + "UID=" + uid + ";" + "PASSWORD=" + pwd + ";" + "Convert Zero Datetime=True;";

//Assigning the connection

mysqlConn = new MySqlConnection(qry);

//open the connection

mysqlConn.Open();

//pass the connection

return mysqlConn;

}

//creating the close connection method

public void CloseConn()

{

//close the connection

mysqlConn.Close();

}

//creating the Data Load Method

public object LoadGrid(string sql)

{

//Assigning the adaptor control command

mysqlAdapt = new MySqlDataAdapter(sql, mysqlConn);

//fill the dataset

mysqlAdapt.Fill(dsDoners = new DataSet());

//Assigning the the values in Grid Control

object dataload = dsDoners.Tables[0];

//pass the values from

return dataload;

}

//create the search record metod

public MySqlDataReader myRead(string sql)

{

//set the connection

mysqlComm = new MySqlCommand(sql, mysqlConn);

//Read the statement

mysqlRead = mysqlComm.ExecuteReader();

//Pass the value

return mysqlRead;

}

//Create the table handling mehod

public void writeRec(string sql)

{

//set the connection in the statement

mysqlComm = new MySqlCommand(sql, mysqlConn);

//update the record

mysqlComm.ExecuteNonQuery();

}

}

}

1. Login Form

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ProjectAlpha

{

public partial class LoginForm : Form

{

public LoginForm()

{

InitializeComponent();

}

private void cle()

{

//Reset all the controls

txtUser.ResetText();

txtPass.ResetText();

}

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

{

//Close the Application

Application.Exit();

}

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

{

//Creating the memory variables

string user, pass;

//Assigning the values to the variables

user = txtUser.Text;

pass = txtPass.Text;

//cheacking the username and password

if (user == "Admin" && pass == "admin")

{

this.Hide();

Main ss = new Main();

ss.Show();

}

else

{

//Displaying the Error Message

MessageBox.Show("Username or Password incorrect!!! Please Check your Username and Passsword and try again","login Error",MessageBoxButtons.OK,MessageBoxIcon.Error);

//Calling the clear procedure

cle();

}

}

}

1. Main Menu Form

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ProjectAlpha

{

public partial class Main : Form

{

public Main()

{

InitializeComponent();

}

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

{

//Open the Doner Registration form

FrmDonerReg formReg = new FrmDonerReg();

formReg.Show();

}

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

{

//close the application

Application.Exit();

}

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

{

//opening the loging form

LoginForm login = new LoginForm();

login.Show();

this.Hide();

}

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

{

//Opening FindDoners form

FindDoners FindDoners = new FindDoners();

FindDoners.Show();

}

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

{

//Open BloodtypesF Form

DonerBloodTypes Bloodtypes = new DonerBloodTypes();

Bloodtypes.Show();

}

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

{

//Open BloodBank stock Form

BloodStock Stock = new BloodStock();

Stock.Show();

}

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

{

//Open Bloodbank StockUpdate Form

UpdateStock US = new UpdateStock();

US.Show();

}

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

{

//Showing Date and time On Form

timer1.Start();

lblTime.Text =DateTime.Now.ToLongTimeString();

lblDate.Text = DateTime.Now.ToLongDateString();

}

private void timer1\_Tick(object sender, EventArgs e)

{

lblTime.Text = DateTime.Now.ToLongTimeString();

timer1.Start();

}

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

{

//open Blood Recipients Form

BloodRecipients bloodRecipients = new BloodRecipients();

bloodRecipients.Show();

}

}

}

1. Doner Registration Form

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ProjectAlpha

{

public partial class FrmDonerReg : Form

{

public FrmDonerReg()

{

InitializeComponent();

}

private void cle()

{

//reset all the controls

txtDID.ResetText();

txtAdd.ResetText();

txtAge.ResetText();

txtDName.ResetText();

txtEmail.ResetText();

txtNIC1.ResetText();

txtTelNo.ResetText();

CmbBloodType.ResetText();

rdoFemale.Checked = false;

rdoMale.Checked = false;

}

//creating memory variable

String gen;

//Inherit the class

DatabaseClass dbClz = new DatabaseClass();

private void button1\_Click(object sender, EventArgs e)

{

//using error handler tool

try

{

//call connection method

dbClz.myCon();

//call write record method

dbClz.writeRec("insert into donerregistration values('" + txtDID.Text + "', '" + txtDName.Text + "', '" + txtNIC1.Text + "', '" + gen + "', '" + txtAdd.Text + "', '" + txtEmail.Text + "', '" + txtAge.Text + "', '" + txtTelNo.Text + "', '" + CmbBloodType.Text + "')");

//show the message

MessageBox.Show("New Record added Suuccessfully", "Donor Registration", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

//Display the Error Message

MessageBox.Show("Error :" + ex.Message, "Donor Registration", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//Call the Load Grid Method

DgvDoner.DataSource = dbClz.LoadGrid("select \* from donerregistration");

//Close he Connection

dbClz.CloseConn();

//Call Clear Method

cle();

}

}

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

{

//Using Error Handler tool

try

{

//open the connection

dbClz.myCon();

//Call the load mthod

DgvDoner.DataSource = dbClz.LoadGrid("select \* from donerregistration");

}

catch (Exception ex)

{

//catch the error

MessageBox.Show("Error : " + ex.Message, "Doner Registration", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//colose the connection

dbClz.CloseConn();

}

}

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

{

//close the register form

this.Hide();

}

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

{

//Call the clear procedure

cle();

}

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

{

//set the values

gen = "Male";

}

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

{

//set the values

gen = "Female";

}

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

{

}

{

}

}

}

1. Doner Details Form

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ProjectAlpha

{

public partial class FindDoners : Form

{

public FindDoners()

{

InitializeComponent();

}

//Inherit the class

DatabaseClass dbClz = new DatabaseClass();

private void cle()

{

//Reset all controls

txtSearch.ResetText();

txtAdd.ResetText();

txtAge.ResetText();

comboBoxBlood.ResetText();

txtDID.ResetText();

txtDName.ResetText();

txtEmail.ResetText();

txtGen.ResetText();

txtNIC.ResetText();

txtTelNo.ResetText();

}

private void enet()

{

txtAdd.Enabled = true;

txtAge.Enabled = true;

comboBoxBlood.Enabled = true;

txtDName.Enabled = true;

txtEmail.Enabled = true;

txtGen.Enabled = true;

txtNIC.Enabled = true;

txtTelNo.Enabled = true;

}

private void enef()

{

txtAdd.Enabled = false;

txtAge.Enabled = false;

comboBoxBlood.Enabled = false;

txtDName.Enabled = false;

txtEmail.Enabled = false;

txtGen.Enabled = false;

txtNIC.Enabled = false;

txtTelNo.Enabled = false;

}

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

{

//using error handler tool

try

{

//open the connection

dbClz.myCon();

//call the load method

dataGridView1.DataSource = dbClz.LoadGrid("select \* from donerregistration");

}

catch(Exception ex)

{

//catch the error

MessageBox.Show("Error: " + ex.Message, "Doner Details", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//close the connection

dbClz.CloseConn();

}

}

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

{

try

{

//call the connection method

dbClz.myCon();

//create the reader and assigning the values

MySql.Data.MySqlClient.MySqlDataReader dr = dbClz.myRead("Select \* from donerregistration where "+comboBoxAccording.Text+"='"+txtSearch.Text+"'");

//using try catch function

try

{

//Read the reader command

dr.Read();

//Assigning the values

txtDID.Text = dr[0].ToString();

txtDName.Text = dr[1].ToString();

txtNIC.Text = dr[2].ToString();

txtGen.Text = dr[3].ToString();

txtAdd.Text = dr[4].ToString();

txtEmail.Text = dr[5].ToString();

txtAge.Text = dr[6].ToString();

txtTelNo.Text = dr[7].ToString();

comboBoxBlood.Text = dr[8].ToString();

//display the message

MessageBox.Show("Record Found !!", "Doner Infomation", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch

{

//catch the error

MessageBox.Show("Record Not Found !!", "Doner Infomation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

//clear the text boxes

txtSearch.Focus();

cle();

}

}

catch(Exception ex)

{

//catch the error

MessageBox.Show("Error : " + ex.Message, "Doner infomation", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

}

}

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

{

//call the clear procedure

cle();

}

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

{

this.Hide();

}

private void FindDoners\_DoubleClick(object sender, EventArgs e)

{

}

private void txtDID\_DoubleClick(object sender, EventArgs e)

{

}

private void txtDID\_Enter(object sender, EventArgs e)

{

}

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

{

if (txtAdd.Enabled == false)

{

//call the enable procedure

enet();

}

else

{

//using the error handler tool

try

{

//call the connection method

dbClz.myCon();

//call write record method

dbClz.writeRec("update donerregistration set DonerName='" + txtDName.Text + "',NIC='" + txtNIC.Text + "',Gender='" + txtGen.Text + "',Address='" + txtAdd.Text + "',Email='" + txtEmail.Text + "',Age='" + txtAge.Text + "',TelNo='" + txtTelNo.Text + "',BloodGroup='" + comboBoxBlood.Text + "'where DonerID='" + txtDID.Text + "'");

//show the message

MessageBox.Show("Record Updated Successfully !!", "Donor Details Updater", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

//Display Error Message

MessageBox.Show("Error : " + ex.Message, "Donor details Updater", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//Call the Load Grid Method

dataGridView1.DataSource = dbClz.LoadGrid("select \* from donerregistration");

//Close he Connection

dbClz.CloseConn();

//Call Clear Method

cle();

//call enable prodedure false

enef();

}

}

}

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

{

btnUpdate.Enabled = !string.IsNullOrEmpty(txtDID.Text);

btnDel.Enabled = !string.IsNullOrEmpty(txtDID.Text);

}

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

{

if (MessageBox.Show("Do you want to Delete the Record?", "Donor Details", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

//using error handler tool

try

{

//call the connection method

dbClz.myCon();

//call th write record method

dbClz.writeRec("delete from donerregistration where DonerID='" + txtDID.Text + "'");

//show the message

MessageBox.Show("Record deleted successfully", "Donor Details", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch(Exception ex)

{

//Display Error Message

MessageBox.Show("Error : " + ex.Message, "Donor Details", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//Call the Load Grid Method

dataGridView1.DataSource = dbClz.LoadGrid("select \* from donerregistration");

//Close he Connection

dbClz.CloseConn();

//Call Clear Method

cle();

//call enable prodedure false

enef();

}

}

}

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

{

//Open the Doner Registration form

FrmDonerReg formReg = new FrmDonerReg();

formReg.Show();

this.Hide();

}

}

}

1. Search a doner through blood Group

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ProjectAlpha

{

public partial class DonerBloodTypes : Form

{

public DonerBloodTypes()

{

InitializeComponent();

}

//Inherit the class

DatabaseClass dbClz = new DatabaseClass();

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

{

this.Hide();

}

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

{

//Using Error Handler tool

try

{

//open the connection

dbClz.myCon();

//Call the load mthod

dgvBloodtypes.DataSource = dbClz.LoadGrid("select \* from donerregistration where BloodGroup='" + comboBoxBloodTypes.Text + "';");

}

catch (Exception ex)

{

//catch the error

MessageBox.Show("Error : " + ex.Message, "Doner Registration", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//colose the connection

dbClz.CloseConn();

}

}

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

{

FindDoners FindDoners = new FindDoners();

FindDoners.Show();

this.Hide();

}

}

}

1. Blood Bank Stock Form

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ProjectAlpha

{

public partial class BloodStock : Form

{

public BloodStock()

{

InitializeComponent();

}

//Inherit the class

DatabaseClass dbClz = new DatabaseClass();

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

{

this.Hide();

UpdateStock US = new UpdateStock();

US.Show();

}

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

{

//Using Error Handler tool

try

{

//open the connection

dbClz.myCon();

//Call the load mthod

dataGridViewStock.DataSource = dbClz.LoadGrid("select \* from bloodstock");

}

catch (Exception ex)

{

//catch the error

MessageBox.Show("Error : " + ex.Message, "Blood Stock Table", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//colose the connection

dbClz.CloseConn();

}

}

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

{

this.Hide();

}

}

}

1. Update Stock Form

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ProjectAlpha

{

public partial class UpdateStock : Form

{

public UpdateStock()

{

InitializeComponent();

}

//Inherit the class

DatabaseClass dbClz = new DatabaseClass();

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

{

//Using Error Handler tool

try

{

//open the connection

dbClz.myCon();

//Call the load mthod

dataGridViewStockUpdate.DataSource = dbClz.LoadGrid("select \* from bloodstock");

}

catch (Exception ex)

{

//catch the error

MessageBox.Show("Error : " + ex.Message, "Blood Stock Update", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//colose the connection

dbClz.CloseConn();

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

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

{

btnUpdate.Enabled = !string.IsNullOrEmpty(txtValue.Text);

}

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

{

//using the error handler tool

try

{

//call the connection method

dbClz.myCon();

//call write record method

dbClz.writeRec("update bloodstock set Available\_Stock='" + txtValue.Text + "'where BloodGroup = '" + comboBox1.Text + "'");

//show the message

MessageBox.Show("Record Updated Successfully !!", "Donor Details Updater", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

//Display Error Message

MessageBox.Show("Error : " + ex.Message, "Blood Stock Updater", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//Call the Load Grid Method

dataGridViewStockUpdate.DataSource = dbClz.LoadGrid("select \* from bloodstock");

//Close he Connection

dbClz.CloseConn();

//Call Clear Method

txtValue.ResetText();

comboBox1.ResetText();

//call enable prodedure false

btnUpdate.Enabled = false;

}

}

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

{

this.Hide();

}

}

}

1. Blood Recipients Form

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ProjectAlpha

{

public partial class BloodRecipients : Form

{

public BloodRecipients()

{

InitializeComponent();

}

//creating memory variable

String gen;

//Inherit the class

DatabaseClass dbClz = new DatabaseClass();

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

{

//Using Error Handler tool

try

{

//open the connection

dbClz.myCon();

//Call the load mthod

dataGridView1.DataSource = dbClz.LoadGrid("select \* from bloodrecipient");

}

catch (Exception ex)

{

//catch the error

MessageBox.Show("Error : " + ex.Message, "bloodrecipient", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//colose the connection

dbClz.CloseConn();

}

}

private void cle()

{

//reset all the controls

txtRID.ResetText();

txtRName.ResetText();

txtNIC1.ResetText();

txtAdd.ResetText();

txtEmail.ResetText();

txtAge.ResetText();

txtTelNo.ResetText();

CmbBloodType.ResetText();

rdoFemale.Checked = false;

rdoMale.Checked = false;

dateTimePicker.Text =String.Empty;

}

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

{

//Call the clear procedure

cle();

txtRID.Focus();

}

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

{

//using error handler tool

try

{

//call connection method

dbClz.myCon();

//call write record method

dbClz.writeRec("INSERT INTO `bloodrecipient` (`RecipientID`, `RecipientName`, `NIC`, `Gender`, `Address`, `Email`, `Age`, `TelNo`, `BloodGroup`, `DateOfReciving`) VALUES ('"+txtRID.Text+"', '"+txtRName.Text+"', '"+txtNIC1.Text+"', '"+gen+"', '"+txtAdd.Text+"', '"+txtEmail.Text+"', '"+txtAge.Text+"', '"+txtTelNo.Text+"', '"+CmbBloodType.Text+"', '"+dateTimePicker.Text+"')");

//show the message

MessageBox.Show("New Record added successfully!", "Blood Recipient", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

//Display the Error Message

MessageBox.Show("Error :" + ex.Message, "Blood Recipient", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//Call the Load Grid Method

dataGridView1.DataSource = dbClz.LoadGrid("select \* from bloodrecipient");

//Close he Connection

dbClz.CloseConn();

//Call Clear Method

cle();

}

}

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

{

//set the values

gen = "Male";

}

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

{

//set the values

gen = "Female";

}

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

{

//close the register form

this.Hide();

}

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

{

try

{

//call the connection method

dbClz.myCon();

//create the reader and assigning the values

MySql.Data.MySqlClient.MySqlDataReader dr = dbClz.myRead("Select \* from bloodrecipient where NIC='" + txtNIC1.Text+"'");

//using try catch function

try

{

//Read the reader command

dr.Read();

//Assigning the values

txtRID.Text = dr[0].ToString();

txtRName.Text = dr[1].ToString();

txtNIC1.Text = dr[2].ToString();

gen = dr[3].ToString();

txtAdd.Text = dr[4].ToString();

txtEmail.Text = dr[5].ToString();

txtAge.Text = dr[6].ToString();

txtTelNo.Text = dr[7].ToString();

CmbBloodType.Text = dr[8].ToString();

dateTimePicker.Text = dr[9].ToString();

//ASsigning radiobutton for gender

if (gen == "Male")

{

rdoMale.Checked = true;

rdoFemale.Checked = false;

}

else

{

rdoMale.Checked = false;

rdoFemale.Checked = true;

}

//display the message

MessageBox.Show("Record Found !!", "Recipient Infomation", MessageBoxButtons.OK, MessageBoxIcon.Information);

//Enable update and delete button

btnUpdate.Enabled = true;

btnDelete.Enabled = true;

btnClear.Enabled = false;

btnSave.Enabled = false;

txtRID.Enabled = false;

}

catch

{

//catch the error

MessageBox.Show("Record Not Found !!", "Recipient Infomation", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

//clear the text boxes

txtNIC1.Focus();

cle();

}

finally

{

}

}

catch (Exception ex)

{

//catch the error

MessageBox.Show("Error : " + ex.Message, "Recipient Infomation", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

}

}

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

{

//using the error handler tool

try

{

//call the connection method

dbClz.myCon();

//call write record method

dbClz.writeRec("update bloodrecipient set RecipientName='" + txtRName.Text + "',NIC='" + txtNIC1.Text + "',Gender='" + gen + "',Address='" + txtAdd.Text + "',Email='" + txtEmail.Text + "',Age='" + txtAge.Text + "',TelNo='" + txtTelNo.Text + "',BloodGroup='" + CmbBloodType.Text + "',DateOfReciving='" +dateTimePicker.Text+ "'where RecipientID='" + txtRID.Text + "'");

//show the message

MessageBox.Show("Record Updated Successfully !!", "Recipient Details Updater", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

//Display Error Message

MessageBox.Show("Error : " + ex.Message, "Recipient Details Updater", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//Call the Load Grid Method

dataGridView1.DataSource = dbClz.LoadGrid("select \* from bloodrecipient");

//Close he Connection

dbClz.CloseConn();

//Call Clear Method

cle();

txtRID.Enabled = true;

//disable update and delete button

btnUpdate.Enabled = false;

btnDelete.Enabled = false;

btnClear.Enabled = true;

btnSave.Enabled = true;

}

}

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

{

if (MessageBox.Show("Do you want to Delete the Record?", "Recipient Details", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

try

{

//call the connection method

dbClz.myCon();

//call th write record method

dbClz.writeRec("delete from bloodrecipient where RecipientID='" + txtRID.Text + "'");

//show the message

MessageBox.Show("Record deleted successfully", "Recipient Details", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

//Display Error Message

MessageBox.Show("Error : " + ex.Message, "Recipient Details", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

//Call the Load Grid Method

dataGridView1.DataSource = dbClz.LoadGrid("select \* from bloodrecipient");

//Close he Connection

dbClz.CloseConn();

//Call Clear Method

cle();

txtRID.Enabled = true;

//disable update and delete button

btnUpdate.Enabled = false;

btnDelete.Enabled = false;

btnClear.Enabled = true;

btnSave.Enabled=true;

}

}

}

}

}

# 6.Software Test

The following procedure was followed when testing the above software

* First each model was followed when tested separately and their flows were fixed.
* In each module testing was mainly focused mainly focused on the underlying section
  + Whether the text fields are coded to expect required data types
  + Assembly with database
  + Whether data gets enter properly to the database
  + Function of buttons Register, clear, return, update, delete, search
* Then the modules are tested as pairs and the errors found were corrected in this the focus was given to the connection between each two models.
* All models were integrated, and the software was tested as a whole, and the defect were developed for the last time in this step. Attention was given to the following areas
  + Connection among the modules of the software
  + The complete process performed by the software
  + Whether software was designed according to the software design
  + Whether the software is user friendly and easy to use
  + Possibility of maintenance of the software
  + Ability to make changes as per the evolving need of the customer in the future
  + Whether the software satisfied the requirements of the customer

1. Unit Testing

Generally, in the blood bank management system needs to be accepted through the unit testing mode. It’s done through the system interface and functions have created. This individual interface was created and tested for the users to be satisfied about the system. This integration test was done to see if the individual programs/ functions were up to expected level of users.

1. System Testing

In the blood bank management system, it was dynamic to run as system test. Using the Blackbox testing method we can check on the result. If the result gets correct, then the system is approved. But it never showed the process at all. But if we Courage The use white box testing method it has a skill to show how

the process going to be performed as well.

1. Test case
   1. Login Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Description | Expected result | Actual Result | Pass/Fail |
| 01 | Run program | Program starts | Program starts | Pass |
| 02 | Enter correct UserName and password and clicking “Login” button | Main Menu form opening | Main Menu form opening | Pass |
| 03 | Enter Incorrect UserName and Password and clicking “Login” button | Message box “login Error” | Message box “login Error” | pass |
| 04 | Clicking “Exit” button | Program ends | Program ends | pass |

* 1. Main Menu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Description | Expected result | Actual Result | Pass/Fail |
| 05 | Click on “Doner Registration” button | Blood Doner Registration form opening | Blood Doner Registration form opening | Pass |
| 06 | Click on “Doner Information” button | Doner Details form opening | Doner Details form opening | Pass |
| 07 | Click on “Search a Doner through Blood Group” | Blood Donors form opening | Blood Donors form opening | Pass |
| 08 | Click on “Blood Bank Stock” | Blood Bank Stock form opening | Blood Bank Stock form opening | Pass |
| 09 | Click on “Update Stock” button | Update Blood Stock form opens | Update Blood Stock form opens | Pass |

* 1. Blood Doner Registration form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Description | Expected result | Actual Result | Pass/Fail |
| 10 | Enter the details correctly & click on Register button | Display message box “New Record added successfully!”, Data grid updates | Display message box “New Record added successfully!”, Data grid updates | Pass |
| 11 | Enter the details incompletely & click on Register button | Display message with the error | Display message with the error | Pass |
| 12 | Click clear button when text boxes filled | All the text fields should be cleared | All the text fields should be cleared | Pass |
| 13 | Click Return To Main Menu button | Return to Main Menu closing Doner registration Form | Return to Main Menu closing Doner registration Form | Pass |

* 1. Doner Details Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Description | Expected result | Actual Result | Pass/Fail |
| 14 | Select “Search According to” and then “Value” According to required search and click Search button and search record available on the database | Display message box “Record Found !!” and all the relevant details should be display on table shown | Display message box “Record Found !!” and all the relevant details should be display on table shown | Pass |
| 14.1 | Click update Record | Enable test field in table to edit | Enable test field in table to edit | Pass |
| 14.2.1 | Change the required fields and click update record button | Display message box “Record Updated Successfully” and update database and update date grid | Display message box “Record Updated Successfully” and update database and update date grid | Pass |
| 14.2 | Click Delete Record button | Display message box “Do you want to Delete the Record” | Display message box “Do you want to Delete the Record” | pass |
| 14.2.1 | Click Yes button | Display message box “Record Deleted successfully” | Display message box “Record Deleted successfully” | Pass |
| 14.2.2 | Click No button | Close message box “Do you want to Delete the Record” | Close message box “Do you want to Delete the Record” | Pass |
| 15 | Click Clear button | Clear all text fields | Clear all text fields | Pass |
| 16 | Click Back Button | Close Doner details form | Close Doner details form | Pass |
| 17 | Click Add a New Doner button | Close Doner details Form and show Blood Doner Registration form | Close Doner details Form and show Blood Doner Registration form | Pass |

* 1. Blood Doners Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Description | Expected result | Actual Result | Pass/Fail |
| 18 | Select a blood Type From combo box | Show all records of the doners with selected blood type on data grid | Show all records of the doners with selected blood type on data grid | Pass |
| 19 | Click Back Button | Close Blood Doners form | Close Blood Doners form | Pass |
| 20 | Click Doner Information button | Close Blood Doners Form and show Blood Doner details form | Close Blood Doners Form and show Blood Doner details form | Pass |

* 1. Blood Bank Stock Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Description | Expected result | Actual Result | Pass/Fail |
| 21 | Click Update Blood Stock button | Show Update Blood Stock Form, close Blood Bank Stock Form | Show Update Blood Stock Form, close Blood Bank Stock Form | Pass |
| 22 | Click Close button | Close blood bank stock form | Close blood bank stock form | Pass |

* 1. Update Stock Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case ID | Description | Expected result | Actual Result | Pass/Fail |
| 23 | Click Close button | Close update Blood stock Form | Close update Blood stock Form | Pass |
| 24 | Select blood group from combo box and enter quantity of available stock | Display message box “Record Updated successfully” and update data grid | Display message box “Record Updated successfully” and update data grid | Pass |

# 7.Grant Chart

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Activity** | 9th of September to 22nd of September | | | | | | | | | | | | | | |
| 1st week | | | | | | | 2nd week | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | Planning |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Requirement Analysis & Definition |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | System & Software Design |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Detailed component Design |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Implementation & Unit Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | Software Integration & System Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | Design Documentation & Typing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 | System Installing & Uses |  |  |  |  |  |  |  |  |  |  |  |  |  |  |