**APPENDIX F: PROGRAM SOURCE** **CODE**

**Program.cs**

using MetroFramework.Controls;

using MySql.Data.MySqlClient;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace PUPiMed

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new LoginNew());

}

public static bool fillComboBox(MetroComboBox comboBox, string strQuery, out List<String> alistCode)

{

comboBox.Items.Clear();

alistCode = new List<String>();

bool retval = true;

using (LoginNew.conn)

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(strQuery, LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

if (dataReader.Read())

{

alistCode.Add(dataReader.GetString(0));

comboBox.Items.Add(dataReader.GetString(1));

while (dataReader.Read())

{

alistCode.Add(dataReader.GetString(0));

comboBox.Items.Add(dataReader.GetString(1));

}

}

else

{

comboBox.Items.Add("");

alistCode.Add("");

retval = false;

}

dataReader.Close();

comboBox.SelectedIndex = 0;

}

}

}

return retval;

}

//returns true if a duplicate is found

public static bool duplicateEntry(DataGridView grid, string value, int column)

{

foreach (DataGridViewRow row in grid.Rows)

{

if (row.Cells[column].Value.ToString().Equals(value))

{

return true;

}

}

return false;

}

public static bool ExecuteQuery(string query)

{

bool result = false;

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

MySqlCommand cmd = new MySqlCommand(query, LoginNew.conn);

if (cmd.ExecuteNonQuery() == 1)

{

result = true;

}

}

catch (Exception ex)

{

MessageBox.Show("\t"+ex.Message.ToString());

}

finally

{

LoginNew.conn.Close();

}

return result;

}

public static string getPrevCode (string query)

{

string prev = null;

try {

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(query, LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

if (dataReader.Read())

{

prev = dataReader.GetString(0);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

return prev;

}

public static string getNextCode(string prev)

{

if (Regex.IsMatch(prev, "^[a-zA-Z0-9-\_]+$"))

{

string original = prev;

string num = string.Empty;

prev = reverse(prev);

char p = '\0', c;

for (int i = 0; i < prev.Length; i++)

{

c = prev[i];

if (Char.IsDigit(c))

{

num += c;

}

else

{

if (Char.IsDigit(p))

break;

}

p = c;

}

if (String.IsNullOrEmpty(num))

{

num = "0";

original += num;

}

num = reverse(num);

ulong nextInt = ulong.Parse(num);

++nextInt;

string newInt = nextInt.ToString();

newInt = reverse(newInt);

if (newInt.Length < num.Length)

{

int i = 0;

int diff = num.Length - newInt.Length;

while (i < diff)

{

newInt += "0";

++i;

}

}

newInt = reverse(newInt);

string nextCode = String.Empty;

int loc = original.LastIndexOf(num);

nextCode = original.Remove(loc, num.Length).Insert(loc, newInt);

return nextCode;

}

return null;

}

public static string reverse(string str)

{

char[] rev = str.ToCharArray();

Array.Reverse(rev);

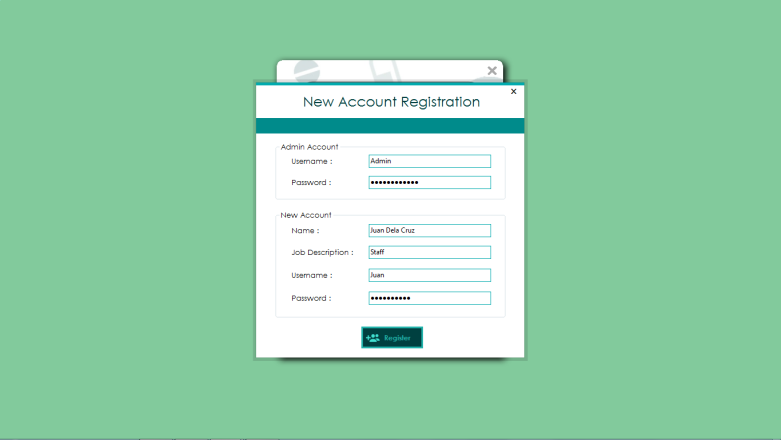
return new String(rev); ;

}

}

}

**New Account Registration**

****

using MetroFramework;

using MetroFramework.Controls;

using MySql.Data.MySqlClient;

using System;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace PUPiMed.Forms

{

public partial class formRegistration : MetroFramework.Forms.MetroForm

{

string server;

public formRegistration(string server)

{

InitializeComponent();

this.server = server;

}

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

{

}

private void button2\_Click(object sender, EventArgs e)

{

this.Dispose();

}

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

{

if (string.IsNullOrEmpty(txtFName.Text) || string.IsNullOrEmpty(txtJob.Text) || string.IsNullOrEmpty(txtUsername.Text) || string.IsNullOrEmpty(txtPassword.Text))

{

error("Please fill out all the fields");

}else

{

try

{

string query =

"CREATE USER '" + txtUsername.Text + "'@'%';" +

"SET PASSWORD FOR '" + txtUsername.Text + "'@'%' = PASSWORD('" + txtPassword.Text + "');" +

"GRANT SELECT, INSERT, EXECUTE, SHOW VIEW, TRIGGER ON dbMedicalClinic.\* TO '" + txtUsername.Text + "'@'%';" +

"INSERT INTO `dbmedicalclinic`.`tbluser` (`strUserUName`, `strUserPassword`, `strUserFirstName`, `strUserMiddleName`, `strUserLastName`, `strUserJob`) VALUES ('" + txtUsername.Text + "', PASSWORD('" + txtPassword.Text + "'), '" + txtFName.Text + "', '" + txtMName.Text + "', '" + txtLName.Text + "', '" + txtJob.Text + "');";

LoginNew.setConnection(txtAUsername.Text, txtAPassword.Text);

if (Program.ExecuteQuery(query))

{

status.BackColor = System.Drawing.Color.LimeGreen;

status.Text = "User " + txtUsername.Text + " successfully created";

foreach (MetroTextBox txtbox in this.groupBox1.Controls.OfType<MetroTextBox>())

{

txtbox.Clear();

}

foreach (MetroTextBox txtbox in this.groupBox2.Controls.OfType<MetroTextBox>())

{

txtbox.Clear();

}

}

}

catch (Exception ex)

{

MetroMessageBox.Show(this, ex.Message, "Access Denied", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

LoginNew.conn.Close();

}

}

}

private void error (string text)

{

status.BackColor = System.Drawing.Color.Tomato;

status.Text = text;

}

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

{

}

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

{

}

}

**Log-in**



using MetroFramework;

using MetroFramework.Forms;

using MySql.Data.MySqlClient;

using System;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

using ToastNotifications;

namespace PUPiMed

{

public partial class LoginNew : MetroForm

{

public static MySqlConnection conn = new MySqlConnection();

public static string server = "localhost";//"192.168.43.112";

public static string user;

public static string pass;

public static string db = "dbmedicalclinic";

public LoginNew()

{

InitializeComponent();

Region = Region.FromHrgn(NativeMethods.CreateRoundRectRgn(0,0, Width, Height, 20, 20));

}

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

{

login(txtUser.Text, txtPass.Text);

}

public void login(string user, string pass)

{

try

{

LoginNew.setConnection(user,pass, txtServer.Text, LoginNew.db);

LoginNew.conn.Open();

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Open))

{

this.txtPass.Clear();

this.txtUser.Clear();

new MainForm(this).Show();

this.Hide();

}

else

{

MetroMessageBox.Show(this, "Access Denied", "Invalid User", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch(Exception ex)

{

MetroMessageBox.Show(this, ex.Message, "Access Denied", MessageBoxButtons.OK ,MessageBoxIcon.Error);

}

finally

{

LoginNew.conn.Close();

}

}

public static void setConnection (string user, string pass)

{

LoginNew.user = user;

LoginNew.pass = pass;

LoginNew.conn.ConnectionString =

"server = " + LoginNew.server + ";" +

"user id =" + LoginNew.user + ";" +

"password=" + LoginNew.pass + ";" +

"database=" + LoginNew.db + ";";

}

public static void setConnection(string user, string pass, string server)

{

LoginNew.user = user;

LoginNew.pass = pass;

LoginNew.server = server;

LoginNew.conn.ConnectionString =

"server = " + LoginNew.server + ";" +

"user id =" + LoginNew.user + ";" +

"password=" + LoginNew.pass + ";" +

"database=" + LoginNew.db + ";";

}

public static void setConnection(string user, string pass, string server, string database)

{

LoginNew.user = user;

LoginNew.pass = pass;

LoginNew.server = server;

LoginNew.db = database;

LoginNew.conn.ConnectionString =

"server = " + LoginNew.server + ";" +

"user id =" + LoginNew.user + ";" +

"password=" + LoginNew.pass + ";" +

"database=" + LoginNew.db + ";";

}

public static string getConnection()

{

return LoginNew.conn.ConnectionString;

}

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

{

this.Dispose();

}

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

{

this.WindowState = FormWindowState.Minimized;

this.Show();

this.WindowState = FormWindowState.Normal;

}

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

{

new Forms.formRegistration(txtServer.Text).ShowDialog();

}

private void txtServer\_MouseDoubleClick(object sender, MouseEventArgs e)

{

(sender as Button).Enabled = true;

}

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

{

LoginNew.server = txtServer.Text;

}

private bool dragging = false;

private Point dragCursorPoint;

private Point dragFormPoint;

private void pictureBox1\_MouseDown(object sender, MouseEventArgs e)

{

dragging = true;

dragCursorPoint = Cursor.Position;

dragFormPoint = this.Location;

}

private void pictureBox1\_MouseMove(object sender, MouseEventArgs e)

{

if (dragging)

{

Point dif = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));

this.Location = Point.Add(dragFormPoint, new Size(dif));

}

}

private void pictureBox1\_MouseUp(object sender, MouseEventArgs e)

{

dragging = false;

}

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

{

}

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

{

}

}

}

**Menu**



using MetroFramework.Controls;

using MetroFramework.Forms;

using MySql.Data.MySqlClient;

using System;

using System.Data;

using System.Windows.Forms;

using System.Drawing;

using System.Linq;

using ToastNotifications;

using PUPiMed;

using PUPiMed.UserControls;

namespace PUPiMed

{

public partial class MainForm : MetroForm

{

LoginNew parent;

private Color backColor1, backColor2, hover, selected;

private void LoadMainForm()

{

this.StartTimer();

this.InitializeComponent();

this.lblTime.ForeColor = System.Drawing.Color.White;

Region = Region.FromHrgn(NativeMethods.CreateRoundRectRgn(0,0, Width, Height, 20, 20));

this.SetStyle(ControlStyles.OptimizedDoubleBuffer, true);

backColor1 = Color.Teal;

backColor2 = Color.LightSeaGreen;

hover = Color.LightCoral;

selected = Color.Firebrick;

changeUC(new UCHome());

txtUser.Text = Program.getPrevCode("SELECT IFNULL((SELECT CONCAT(strUserFirstName, ' ', strUserLastName, ' - ',strUserJob) FROM tblUser WHERE strUserUName LIKE '"+LoginNew.user+"' AND strUserPassword LIKE PASSWORD('"+LoginNew.pass+"')), '"+LoginNew.user+"');");

}

public MainForm(LoginNew parent)

{

this.parent = parent;

this.LoadMainForm();

//this.showExpiredNotif();

}

public MainForm()

{

LoadMainForm();

}

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

{

}

public void showExpiredNotif()

{

MySqlConnection notifConn = LoginNew.conn;

string query = "CALL notif();";

try

{

if (notifConn != null && (notifConn.State == ConnectionState.Closed))

notifConn.Open();

using (MySqlCommand cmd = new MySqlCommand(query, notifConn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

//Read the data and store them in the list

while (dataReader.Read())

{

ToastNotifications.Notification notif = new ToastNotifications.Notification(dataReader[0].ToString(), dataReader[1].ToString(), 7, ToastNotifications.FormAnimator.AnimationMethod.Slide, ToastNotifications.FormAnimator.AnimationDirection.Up);

notif.Show();

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Notif: " + ex.Message);

}

finally

{

notifConn.Close();

}

}

private void resetColor(object sender)

{

try

{

MetroTile mtile = new MetroTile();

mtile = sender as MetroTile;

foreach (var tile in mpLeftPanel.Controls.OfType<MetroTile>())

{

if (tile.Name != "mtLogout"&& tile.TabIndex%2==0)

tile.BackColor = backColor2;

else if (tile.Name != "mtLogout" && tile.TabIndex % 2 != 0)

tile.BackColor = backColor1;

}

mtile.BackColor = selected;

}

catch { }

}

private void StartTimer()

{

System.Windows.Forms.Timer timer1 = new System.Windows.Forms.Timer();

timer1.Interval = 1000;

timer1.Tick += new EventHandler(timer1\_Tick);

timer1.Enabled = true;

}

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

{

lblTime.ForeColor = System.Drawing.Color.White;

lblTime.Text = DateTime.Now.ToLongDateString() +" "+ DateTime.Now.ToLongTimeString();

}

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

{

}

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

{

changeUC(new UCItem(1));

resetColor(sender);

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

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

{

changeUC(new UCFacultyTab());

resetColor(sender);

}

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

{

changeUC(new UCItem(2));

resetColor(sender);

}

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

{

changeUC(new UCItem(3));

resetColor(sender);

}

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

{

changeUC(new UCItemDistribution());

resetColor(sender);

}

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

{

changeUC(new UCAdminTab());

resetColor(sender);

}

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

{

}

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

{

this.parent.Show();

this.Dispose();

}

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

{

changeUC(new UCItemInventory());

resetColor(sender);

}

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

{

changeUC(new UCPatientLogs\_DTR\_());

resetColor(sender);

}

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

{

changeUC(new UCItemLibrary());

resetColor(sender);

}

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

{

}

private void label1\_Click(object sender, EventArgs e)

{

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

}

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

{

changeUC(new ucNotifications());

resetColor(sender);

}

private void MainForm\_MouseEnter(object sender, EventArgs e)

{

}

private void MainForm\_Activated(object sender, EventArgs e)

{

}

private void MainForm\_Shown(object sender, EventArgs e)

{

}

public void changeUC (UserControl uc)

{

try

{

Control ctrlOnScreen = uc;

Control ctrlOffScreen = this.mpanel.Controls[0];

this.mpanel.Controls.Add(ctrlOnScreen);

ctrlOnScreen.BringToFront();

ctrlOffScreen.SendToBack();

this.Controls.Remove(ctrlOffScreen);

}

catch { }

}

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

{

foreach (var tile in mpLeftPanel.Controls.OfType<MetroTile>())

{

if (tile.Name != "mtLogout" && tile.TabIndex % 2 == 0)

tile.BackColor = backColor2;

else if (tile.Name != "mtLogout" && tile.TabIndex % 2 != 0)

tile.BackColor = backColor1;

}

changeUC(new UCHome());

}

private void metroTile1\_Click(object sender, EventArgs e)

{

changeUC(new UCReceive());

resetColor(sender);

}

private void mouseEnter(object sender, EventArgs e)

{

try

{

MetroTile mtile = new MetroTile();

mtile = sender as MetroTile;

if (mtile.BackColor != selected)

mtile.BackColor = hover;

}

catch { }

}

private void mouseExit(object sender, EventArgs e)

{

try

{

MetroTile tile = sender as MetroTile;

if (tile.BackColor != selected)

{

if (tile.TabIndex % 2 == 0)

tile.BackColor = backColor2;

else if (tile.TabIndex % 2 != 0)

tile.BackColor = backColor1;

}

}

catch { }

}

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

{

UCMenuMaintenance uc = new UCMenuMaintenance(this);

showMenu(uc, mtMaintenance.Location.X + this.Location.X + mtMaintenance.Width, mtMaintenance.Location.Y + this.Location.Y -2);

resetColor(sender);

}

private void metroTile2\_Click(object sender, EventArgs e)

{

UCReportSubMenu uc = new UCReportSubMenu(this);

showMenu(uc, metroTile2.Location.X + this.Location.X + metroTile2.Width, (metroTile2.Height + metroTile2.Location.Y+ this.Location.Y) - uc.Height - 3);

resetColor(sender);

}

private void toolTip1\_Popup(object sender, PopupEventArgs e)

{

}

public void showMenu(UserControl uc, int x, int y)

{

formMenu menu = new formMenu(uc);

menu.Location = new Point(x, y);

menu.Show();

menu.FormClosing += Menu\_FormClosing;

menu.Focus();

}

private void Menu\_FormClosing(object sender, FormClosingEventArgs e)

{

this.Focus();

}

private bool dragging = false;

private Point dragCursorPoint;

private Point dragFormPoint;

private void pictureBox1\_MouseDown(object sender, MouseEventArgs e)

{

dragging = true;

dragCursorPoint = Cursor.Position;

dragFormPoint = this.Location;

}

private void pictureBox1\_MouseMove(object sender, MouseEventArgs e)

{

if (dragging)

{

Point dif = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));

this.Location = Point.Add(dragFormPoint, new Size(dif));

}

}

private void pictureBox1\_MouseUp(object sender, MouseEventArgs e)

{

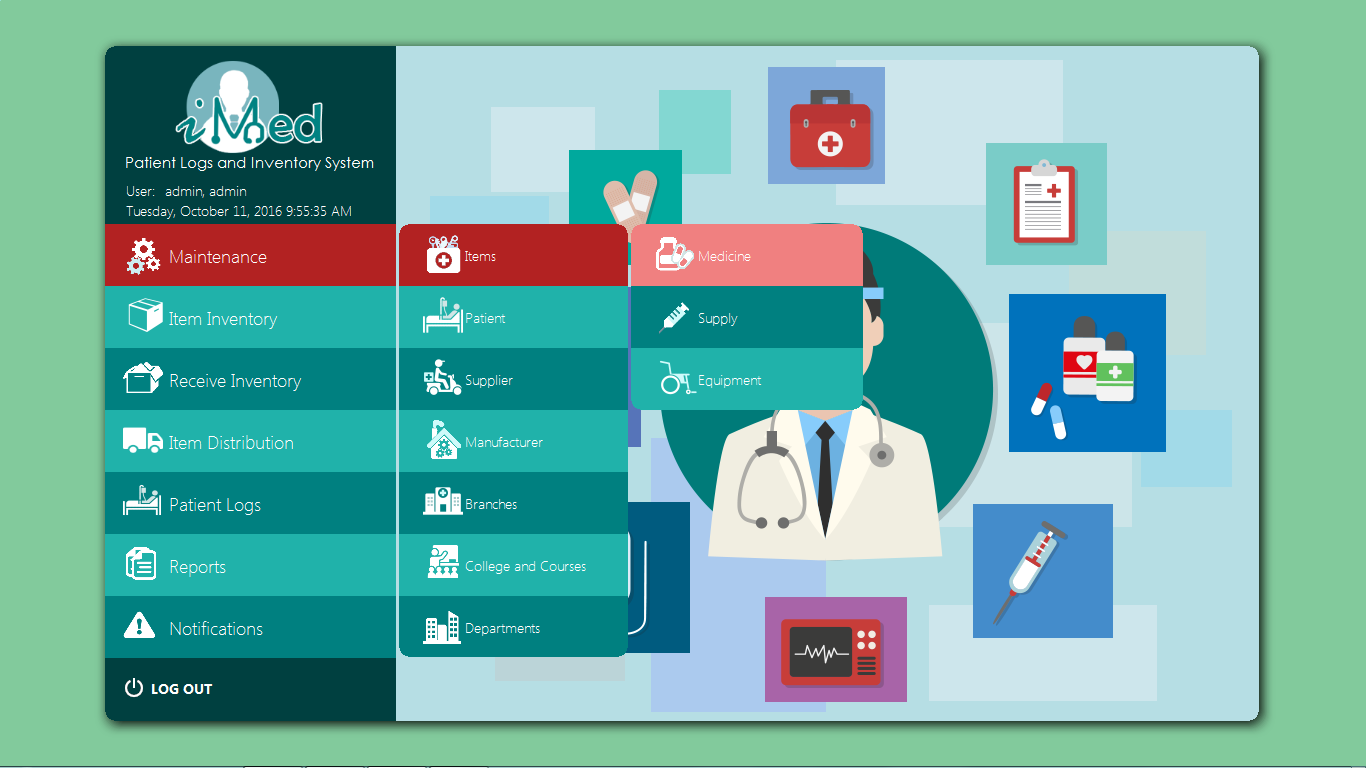
dragging = false;

}

}

}

**Maintenance Menu**

****

using System;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

using PUPiMed.UserControls;

using MetroFramework.Controls;

namespace PUPiMed

{

public partial class UCMenuMaintenance : UserControl

{

private MainForm mainForm;

private formMenu menu;

private Color backColor1, backColor2, hover, selected;

public UCMenuMaintenance(MainForm mainForm)

{

InitializeComponent();

this.mainForm = mainForm;

}

public UCMenuMaintenance()

{

InitializeComponent();

}

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

{

menu = ((formMenu)this.ParentForm);

menu.FormClosing += new FormClosingEventHandler(menu\_FormClosing);

backColor1 = Color.Teal;

backColor2 = Color.LightSeaGreen;

hover = Color.LightCoral;

selected = Color.Firebrick;

}

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

{

try

{

resetColor(sender);

formMenu menu = ((formMenu)this.ParentForm);

menu.deact = false;

UCMenuItems uc = new UCMenuItems(mainForm, menu);

mainForm.showMenu(uc, ParentForm.Location.X + ParentForm.Width, ParentForm.Location.Y);

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

try

{

resetColor(sender);

formMenu menu = ((formMenu)this.ParentForm);

menu.deact = false;

UCMenuPatient uc = new UCMenuPatient(mainForm, menu);

mainForm.showMenu(uc, ParentForm.Location.X + ParentForm.Width, ParentForm.Location.Y + this.tabPatient.Location.Y);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

resetColor(sender);

mainForm.changeUC(new UCSupplier());

menu.Close();

}

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

{

resetColor(sender);

mainForm.changeUC(new UCManufacturer());

menu.Close();

}

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

{

resetColor(sender);

mainForm.changeUC(new UCBranches());

menu.Close();

}

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

{

resetColor(sender);

mainForm.changeUC(new UCCollege());

menu.Close();

}

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

{

resetColor(sender);

mainForm.changeUC(new UCBranches());

menu.Close();

}

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

{

resetColor(sender);

mainForm.changeUC(new UCCollege());

menu.Close();

}

private void metroTile1\_Click(object sender, EventArgs e)

{

resetColor(sender);

mainForm.changeUC(new UserControls.UCDepartment());

menu.Close();

}

private void tabSupply\_MouseEnter(object sender, EventArgs e)

{

menu.active = true;

try

{

MetroTile mtile = new MetroTile();

mtile = sender as MetroTile;

if (mtile.BackColor != selected)

mtile.BackColor = hover;

}

catch { }

}

private void tabSupply\_MouseLeave(object sender, EventArgs e)

{

try

{

menu.active = false;

MetroTile tile = sender as MetroTile;

if (tile.BackColor != selected)

{

if (tile.TabIndex % 2 == 0)

tile.BackColor = backColor2;

else if (tile.TabIndex % 2 != 0)

tile.BackColor = backColor1;

}

}

catch { }

}

private void resetColor(object sender)

{

try

{

MetroTile mtile = new MetroTile();

mtile = sender as MetroTile;

foreach (var tile in this.Controls.OfType<MetroTile>())

{

if (tile.TabIndex % 2 == 0)

tile.BackColor = backColor2;

else if (tile.TabIndex % 2 != 0)

tile.BackColor = backColor1;

}

mtile.BackColor = selected;

}

catch { }

}

private void UCMenuMaintenance\_MouseEnter(object sender, EventArgs e)

{

menu.active = true;

}

private void UCMenuMaintenance\_MouseLeave(object sender, EventArgs e)

{

menu.active = false;

}

private void menu\_FormClosing (object sender, FormClosingEventArgs ex)

{

this.mainForm.Focus();

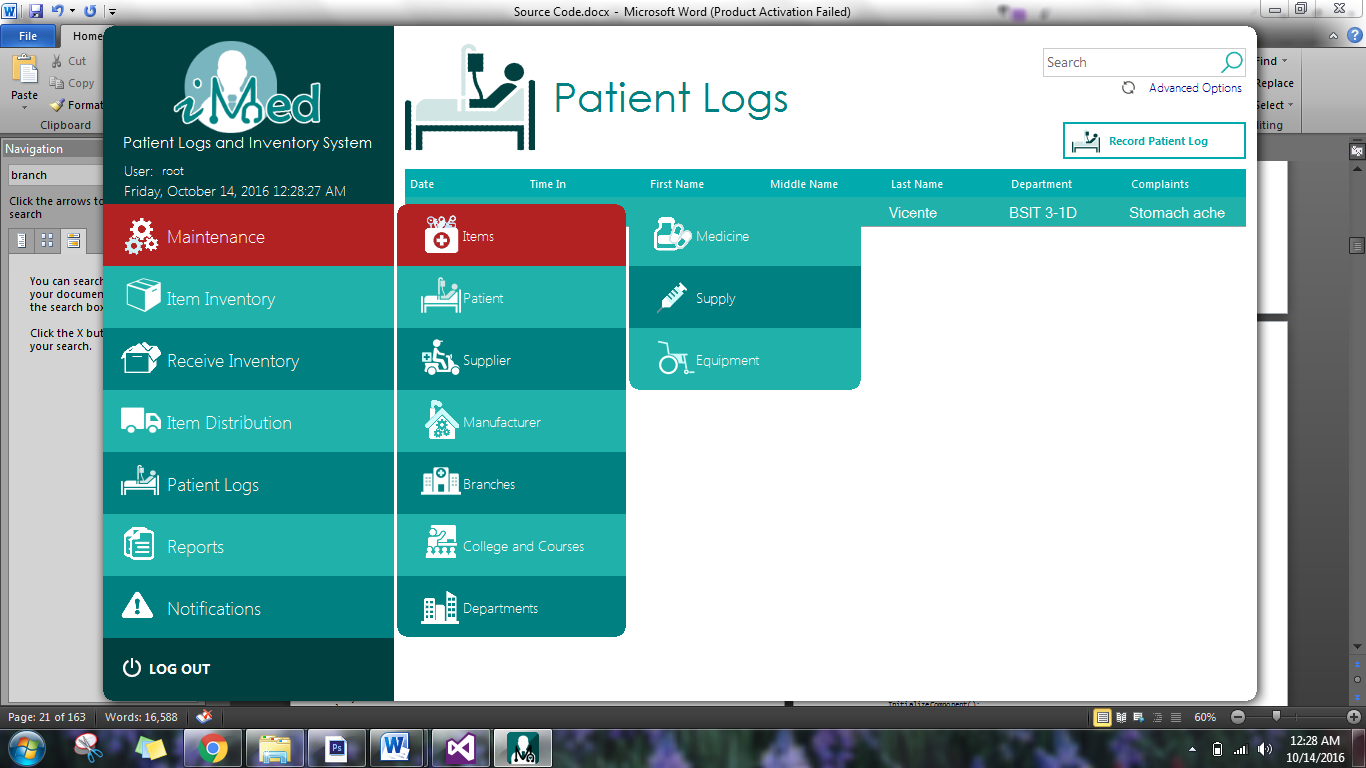
this.mainForm.Activate();

}

}

}

**Items Menu**



using MetroFramework.Controls;

using System;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace PUPiMed

{

public partial class UCMenuItems : UserControl

{

private MainForm mainForm;

private formMenu menu;

private Color backColor1, backColor2, hover, selected;

public UCMenuItems(MainForm mainForm, formMenu menu)

{

InitializeComponent();

this.mainForm = mainForm;

this.menu = menu;

}

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

{

this.ParentForm.FormClosed += new FormClosedEventHandler(this.menu\_FormClosed);

backColor1 = Color.Teal;

backColor2 = Color.LightSeaGreen;

hover = Color.LightCoral;

selected = Color.Firebrick;

}

private void menu\_FormClosed(object sender, FormClosedEventArgs e)

{

if (!this.menu.active)

{

this.menu.Close();

this.mainForm.Focus();

this.mainForm.Activate();

}

else

{

this.menu.Focus();

this.menu.Activate();

}

}

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

{

resetColor(sender);

mainForm.changeUC(new UCItem(1));

ParentForm.Close();

}

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

{

resetColor(sender);

mainForm.changeUC(new UCItem(2));

ParentForm.Close();

}

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

{

resetColor(sender);

mainForm.changeUC(new UCItem(3));

ParentForm.Close();

}

private void tabSupply\_MouseEnter(object sender, EventArgs e)

{

try

{

MetroTile mtile = new MetroTile();

mtile = sender as MetroTile;

if (mtile.BackColor != selected)

mtile.BackColor = hover;

}

catch { }

}

private void tabSupply\_MouseLeave(object sender, EventArgs e)

{

try

{

MetroTile tile = sender as MetroTile;

if (tile.BackColor != selected)

{

if (tile.TabIndex % 2 == 0)

tile.BackColor = backColor2;

else if (tile.TabIndex % 2 != 0)

tile.BackColor = backColor1;

}

}

catch { }

}

private void resetColor(object sender)

{

try

{

MetroTile mtile = new MetroTile();

mtile = sender as MetroTile;

foreach (var tile in this.Controls.OfType<MetroTile>())

{

if (tile.TabIndex % 2 == 0)

tile.BackColor = backColor2;

else if (tile.TabIndex % 2 != 0)

tile.BackColor = backColor1;

}

mtile.BackColor = selected;

}

catch { }

}

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

{

resetColor(sender);

mainForm.changeUC(new UCItemLibrary());

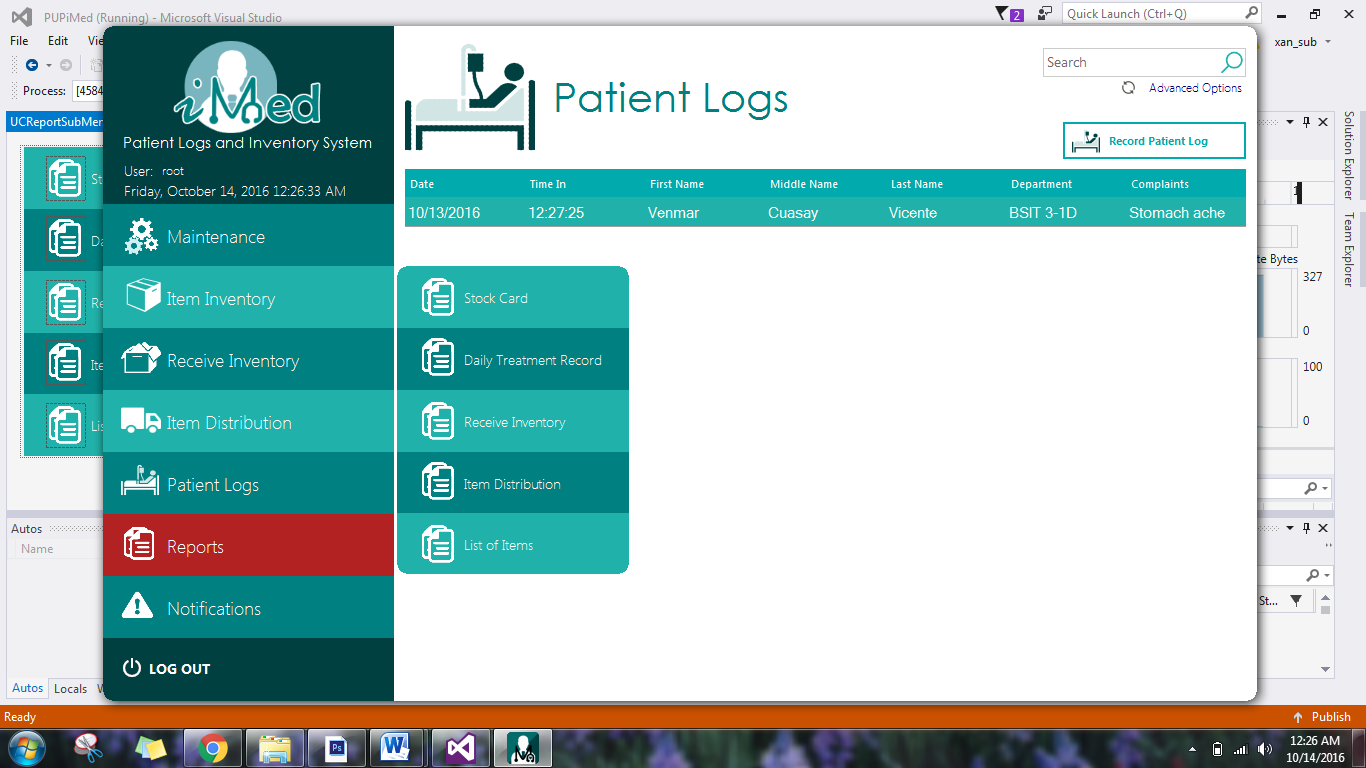
ParentForm.Close();

}

}

}

**Reports Menu**



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MetroFramework.Controls;

namespace PUPiMed.UserControls

{

public partial class UCReportSubMenu : UserControl

{

private MainForm mainForm;

private formMenu menu;

private Color backColor1, backColor2, hover, selected;

public UCReportSubMenu(MainForm mainForm)

{

InitializeComponent();

this.mainForm = mainForm;

}

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

{

menu = ((formMenu)this.ParentForm);

menu.FormClosing += new FormClosingEventHandler(menu\_FormClosing);

backColor1 = Color.Teal;

backColor2 = Color.LightSeaGreen;

hover = Color.LightCoral;

selected = Color.Firebrick;

}

private void menu\_FormClosing(object sender, FormClosingEventArgs ex)

{

this.mainForm.Focus();

this.mainForm.Activate();

}

private void resetColor(object sender)

{

try

{

MetroTile mtile = sender as MetroTile;

foreach (var tile in this.Controls.OfType<MetroTile>())

{

if (tile.TabIndex % 2 == 0)

tile.BackColor = backColor2;

else if (tile.TabIndex % 2 != 0)

tile.BackColor = backColor1;

}

mtile.BackColor = selected;

}

catch { }

}

private void menu\_FormClosed(object sender, FormClosedEventArgs e)

{

if (!this.menu.active)

{

this.menu.Close();

this.mainForm.Focus();

this.mainForm.Activate();

}

else

{

this.menu.Focus();

this.menu.Activate();

}

}

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

{

resetColor(sender);

mainForm.changeUC(new UCReportStockCard());

menu.Close();

}

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

{

resetColor(sender);

mainForm.changeUC(new UCReports());

menu.Close();

}

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

{

resetColor(sender);

mainForm.changeUC(new UCReportReceive());

menu.Close();

}

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

{

resetColor(sender);

mainForm.changeUC(new UCReportDistrib());

menu.Close();

}

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

{

resetColor(sender);

mainForm.changeUC(new UCReportItemList());

menu.Close();

}

private void tabStockCard\_MouseEnter(object sender, EventArgs e)

{

try

{

MetroTile mtile = new MetroTile();

mtile = sender as MetroTile;

if (mtile.BackColor != selected)

mtile.BackColor = hover;

}

catch { }

}

private void tabSupply\_MouseLeave(object sender, EventArgs e)

{

try

{

menu.active = false;

MetroTile tile = sender as MetroTile;

if (tile.BackColor != selected)

{

if (tile.TabIndex % 2 == 0)

tile.BackColor = backColor2;

else if (tile.TabIndex % 2 != 0)

tile.BackColor = backColor1;

}

}

catch { }

}

private void UCReportSubMenu\_MouseEnter(object sender, EventArgs e)

{

menu.active = true;

}

private void UCReportSubMenu\_MouseLeave(object sender, EventArgs e)

{

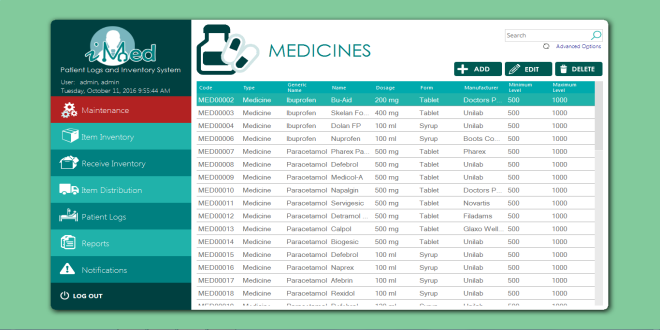
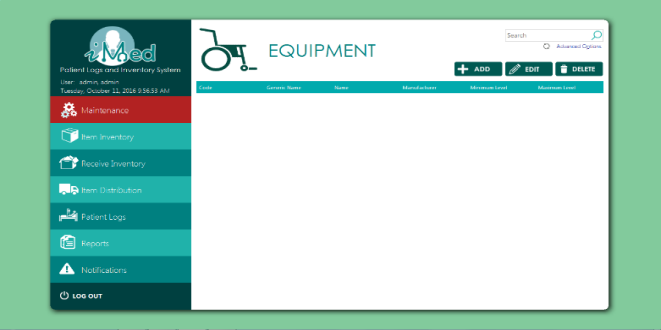
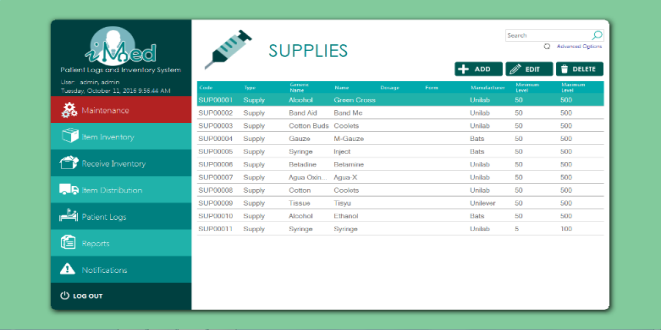
menu.active = false;

}

}

}

**Item Forms**

******

using System;

using System.Data;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using MetroFramework;

namespace PUPiMed

{

public partial class UCItem : UserControl

{

public int itemType;

public string strItem;

public string query;

FormAddItem itemForm;

public UCItem(int itemType)

{

InitializeComponent();

this.itemType = itemType;

}

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

{

switch (this.itemType)

{

case 1:

this.icon.Image = global::PUPiMed.Properties.Resources.mecidine\_teal;

title.Text = "MEDICINES";

strItem = "Medicine";

break;

case 2:

this.icon.Image = global::PUPiMed.Properties.Resources.suppliesdff;

title.Text = "SUPPLIES";

strItem = "Supply";

break;

case 3:

this.icon.Image = global::PUPiMed.Properties.Resources.wheelchair\_teal;

title.Text = "EQUIPMENT";

strItem = "Equipment";

break;

}

filter();

}

public void updateTable()

{

using (LoginNew.conn)

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(query, LoginNew.conn))

{

cmd.CommandType = CommandType.Text;

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

using (DataTable dt = new DataTable())

{

if (dt != null)

{

sda.Fill(dt);

grid.DataSource = dt;

}

}

}

}

}

}

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

{

if(grid.CurrentRow!=null)

{

itemForm = new FormAddItem(this, itemType);

getValues();

itemForm.ShowDialog();

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void grid\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (grid.CurrentRow != null)

{

itemForm = new FormAddItem(this, itemType);

getValues();

itemForm.medicine.Enabled = false;

itemForm.equipment.Enabled = false;

itemForm.ShowDialog();

}

}

private void getValues()

{

itemForm.choice = 1;

if (this.itemType == 3)

{

itemForm.txtCode.Text = grid.CurrentRow.Cells[0].Value.ToString();

itemForm.txtGeneric\_.Text = grid.CurrentRow.Cells[1].Value.ToString();

itemForm.txtName\_.Text = grid.CurrentRow.Cells[2].Value.ToString();

itemForm.cbManufacturer\_.SelectedIndex = itemForm.cbManufacturer\_.Items.IndexOf(grid.CurrentRow.Cells[3].Value.ToString());

itemForm.txtMin\_.Text = grid.CurrentRow.Cells[4].Value.ToString();

itemForm.txtMax\_.Text = grid.CurrentRow.Cells[5].Value.ToString();

}

else

{

itemForm.txtCode.Text = grid.CurrentRow.Cells[0].Value.ToString();

itemForm.txtGen.Text = grid.CurrentRow.Cells[2].Value.ToString();

itemForm.txtName.Text = grid.CurrentRow.Cells[3].Value.ToString();

itemForm.txtDosage.Text = grid.CurrentRow.Cells[4].Value.ToString();

itemForm.cbForm.SelectedIndex = itemForm.cbForm.Items.IndexOf(grid.CurrentRow.Cells[5].Value.ToString());

itemForm.cbManufacturer.SelectedIndex = itemForm.cbManufacturer.Items.IndexOf(grid.CurrentRow.Cells[6].Value.ToString());

itemForm.txtMin.Text = grid.CurrentRow.Cells[7].Value.ToString();

itemForm.txtMax.Text = grid.CurrentRow.Cells[8].Value.ToString();

}

}

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

{

itemForm = new FormAddItem(this, itemType);

itemForm.choice = 0;

itemForm.ShowDialog();

}

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

{

if (grid.CurrentRow != null)

{

var result = MetroMessageBox.Show(this.Parent.Parent, "Are you sure?", "DELETE CONFIRMATION", MessageBoxButtons.YesNo, MessageBoxIcon.Error);

if (result == DialogResult.Yes)

{

string strQuery;

string col1;

if (grid.CurrentRow.Cells[0] != null)

{

col1 = grid.CurrentRow.Cells[0].Value.ToString();

strQuery = "UPDATE tblItem SET boolItemDeleted=1, dtmItemDeleted=NOW(), strItemDeleted=CURRENT\_USER WHERE strItemCode = '" + col1 + "';";

if (Program.ExecuteQuery(strQuery))

{

//success

filter();

}

}

}

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

filter();

}

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

{

FormItemSearch form = new FormItemSearch(this, itemType);

if (itemType == 3)

query = "SELECT `Code`, `Generic Name`, `Name`, `Manufacturer`, `Minimum Level`, `Maximum Level` FROM vwitem WHERE `Type` LIKE '" + strItem + "' AND `Name` LIKE '" + txtFilter.Text + "%' ";

else

query = "SELECT \* FROM vwitem WHERE `Type` LIKE '" + strItem + "' AND `Name` LIKE '" + txtFilter.Text + "%' ";

form.ShowDialog();

}

private void filter()

{

if(itemType == 3)

query = query = "SELECT `Code`, `Generic Name`, `Name`, `Manufacturer`, `Minimum Level`, `Maximum Level` FROM vwitem WHERE `Type` LIKE '" + strItem + "' AND `Name` LIKE '" + txtFilter.Text + "%';";

else

query = "SELECT \* FROM vwitem WHERE `Type` LIKE '" + strItem + "' AND `Name` LIKE '" + txtFilter.Text + "%';";

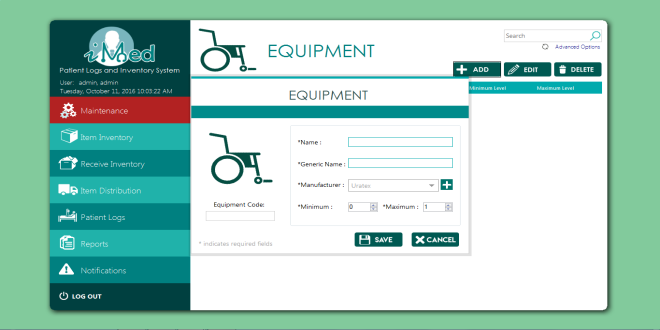
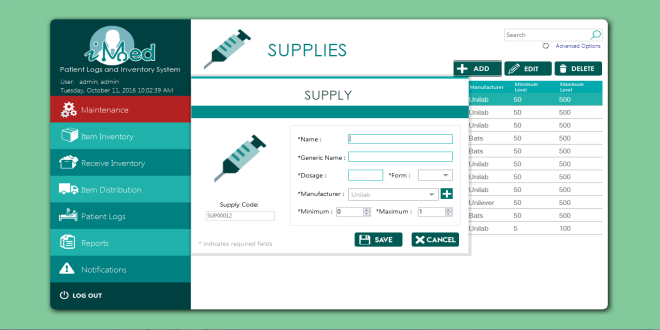
updateTable();

}

}

}

**Add Item**

****

using MetroFramework.Forms;

using PUPiMed.Forms;

using System;

using System.Collections.Generic;

using System.Drawing;

namespace PUPiMed

{

public partial class FormAddItem : MetroForm

{

public UCItem parent;

public int choice = 0, itemType = 1;

List<String> listManuCode;

string strCode, strName, strGen, strManu;

int intMin, intMax;

private UCItemLibrary uCItemLibrary;

public FormAddItem(UCItem parent, int itemType)

{

InitializeComponent();

this.itemType = itemType;

this.parent = parent;

loadItemType();

loadManufacturer();

}

public FormAddItem(int itemType)

{

InitializeComponent();

this.itemType = itemType;

this.choice = 0;

loadItemType();

loadManufacturer();

}

public FormAddItem(UCItemLibrary uCItemLibrary, int itemType)

{

InitializeComponent();

this.uCItemLibrary = uCItemLibrary;

this.itemType = itemType;

this.choice = 1;

loadItemType();

loadManufacturer();

}

public void loadItemType()

{

switch (this.itemType)

{

case 1:

this.icon.Image = global::PUPiMed.Properties.Resources.pill\_teal;

Title.Text = "MEDICINE";

lblCode.Text = "Medicine Code: ";

medicine.Enabled = true;

medicine.Visible = true;

medicine.BringToFront();

equipment.Enabled = false;

equipment.Visible = false;

break;

case 2:

this.icon.Image = global::PUPiMed.Properties.Resources.suppliesdff;

Title.Text = "SUPPLY";

lblCode.Text = "Supply Code: ";

medicine.Enabled = true;

medicine.Visible = true;

medicine.BringToFront();

equipment.Enabled = false;

equipment.Visible = false;

break;

case 3:

this.icon.Image = global::PUPiMed.Properties.Resources.wheelchair\_teal;

Title.Text = "EQUIPMENT";

lblCode.Text = "Equipment Code: ";

medicine.Enabled = false;

medicine.Visible = false;

equipment.BringToFront();

equipment.Enabled = true;

equipment.Visible = true;

break;

}

}

private bool everythingIsOkay()

{

if (this.itemType != 3)

{

strCode = txtCode.Text;

strName = txtName.Text;

strGen = txtGen.Text;

strManu = listManuCode[cbManufacturer.SelectedIndex].ToString();

if (itemType == 1)

{

if (string.IsNullOrEmpty(txtDosage.Text))

{

return false;

}

}

if (!(Int32.TryParse(txtMin.Text, out intMin)))

{

return false;

}

if (!(Int32.TryParse(txtMax.Text, out intMax)))

{

return false;

}

}

else

{

strCode = txtCode.Text;

strName = txtName\_.Text;

strGen = txtGeneric\_.Text;

strManu = listManuCode[cbManufacturer\_.SelectedIndex].ToString();

if (!(Int32.TryParse(txtMin\_.Text, out intMin)))

{

return false;

}

if (!(Int32.TryParse(txtMax\_.Text, out intMax)))

{

return false;

}

}

if (string.IsNullOrEmpty(strCode) || string.IsNullOrEmpty(strName) || string.IsNullOrEmpty(strGen) || string.IsNullOrEmpty(strManu))

{

return false;

}

return true;

}

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

{

string strQuery;

if (everythingIsOkay())

{

if (intMin >= intMax)

{

status.Text = "Max must be greater than the minimum value.";

status.BackColor = Color.Tomato;

}else

{

if (string.IsNullOrEmpty(PUPiMed.Program.getNextCode(strCode)))

{

txtCode.Focus();

status.Text = "Invalid code.";

status.BackColor = Color.Tomato;

}

else

{

if (choice == 0)

{

strQuery = "INSERT INTO tblItem (`strItemCode`, `strItemName`, `strItemGeneric`, `strItemManuCode`, `intItemMin`, `intItemMax`, `intItemType`, `strItemAdded`) VALUES ('" + strCode + "', '" + strName + "', '" + strGen + "', '" + strManu + "', " + intMin + ", " + intMax + ", " + itemType + ", USER());";

}

else

{

strQuery = "UPDATE tblItem SET strItemName='" + strName

+ "', strItemGeneric='" + strGen

+ "', strItemManuCode='" + strManu

+ "', intItemMin=" + intMin

+ ", intItemMax=" + intMax

+ " WHERE strItemCode='" + strCode + "';";

}

if (PUPiMed.Program.ExecuteQuery(strQuery))

{

if (itemType != 3)

{

if (!string.IsNullOrEmpty(txtDosage.Text))

{

string form = cbForm.SelectedItem.ToString();

string query= "";

if (choice == 1)

query = "UPDATE `dbmedicalclinic`.`tbladditional` SET `strAddiDosage`='"+txtDosage.Text+"', `strAddiForm`='"+form+"' WHERE `strAddiItemCode`='"+strCode+"';";

else

query = "INSERT INTO `dbmedicalclinic`.`tbladditional` (`strAddiItemCode`, `strAddiDosage`, `strAddiForm`) VALUES ('" + strCode + "', '" + txtDosage.Text + "', '" + form + "');";

if (Program.ExecuteQuery(query)) { }

//System.Windows.Forms.MessageBox.Show(query);

}

}

//success

loadManufacturer();

strCode = PUPiMed.Program.getNextCode(strCode);

txtCode.Text = strCode;

txtCode.Enabled = false;

if (this.itemType == 3)

{

if (cbManufacturer\_.Items != null)

cbManufacturer\_.SelectedIndex = 0;

txtName\_.Clear();

txtName\_.Focus();

txtGeneric\_.Clear();

txtMin\_.Value = txtMin\_.Minimum;

txtMax\_.Value = txtMax\_.Minimum;

}

else

{

if (cbManufacturer.Items != null)

cbManufacturer.SelectedIndex = 0;

txtName.Clear();

txtName.Focus();

txtGen.Clear();

txtMin.Value = txtMin.Minimum;

txtMax.Value = txtMax.Maximum;

txtDosage.Clear();

cbForm.SelectedIndex = 0;

}

if (parent != null)

parent.updateTable();

if (this.uCItemLibrary != null)

uCItemLibrary.updateTable();

status.Text = "Saved";

status.BackColor = Color.LimeGreen;

if (choice == 1)

this.Dispose();

}

else

{

//fail

status.Text = "Failed to make changes";

status.BackColor = Color.Tomato;

}

}

}

}

else

{

//tell them it's not :(

status.Text = "Please fill out all the required fields";

status.BackColor = Color.Tomato;

}

}

public void loadManufacturer()

{

//Meds & Supplies Only

if (this.itemType != 3)

{

if (Program.fillComboBox(cbManufacturer, "SELECT \* FROM tblManufacturer WHERE boolManuDeleted = false AND intManuType=" + itemType + ";", out listManuCode))

{

cbManufacturer.PromptText = "Please add a new Manufacturer";

}

}

//Equipment Only

else

{

if (Program.fillComboBox(cbManufacturer\_, "SELECT \* FROM tblManufacturer WHERE boolManuDeleted = false AND intManuType=" + itemType + ";", out listManuCode))

{

cbManufacturer\_.PromptText = "Please add a new Manufacturer";

}

}

}

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

{

if (choice == 1)

{

txtCode.Enabled = false;

txtName.Focus();

}

else

{

getCode();

}

cbForm.SelectedIndex = 0;

}

private void txtMin\_KeyPress(object sender, System.Windows.Forms.KeyPressEventArgs e)

{

e.Handled = !char.IsDigit(e.KeyC­har) && !char.IsControl(e.Ke­yChar);

}

private void txtMax\_KeyPress(object sender, System.Windows.Forms.KeyPressEventArgs e)

{

e.Handled = !char.IsDigit(e.KeyC­har) && !char.IsControl(e.Ke­yChar);

}

private void txtMin\_ValueChanged(object sender, EventArgs e)

{

txtMin.Minimum = 0;

}

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

{

new FormAddManufacturer(this, this.itemType-1).ShowDialog();

}

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

{

}

private void txtMax\_ValueChanged(object sender, EventArgs e)

{

}

private void getCode()

{

//check if theres an existing code

string prevCode = PUPiMed.Program.getPrevCode("SELECT strItemCode from tblItem WHERE intItemType=" + itemType + " ORDER by CHAR\_LENGTH(strItemCode) DESC, strItemCode DESC LIMIT 1;");

if (string.IsNullOrEmpty(prevCode))

{

this.txtCode.Enabled = true;

this.txtCode.Focus();

}

else

{

strCode = PUPiMed.Program.getNextCode(prevCode);

this.txtCode.Text = strCode;

this.txtCode.Enabled = false;

this.txtName.Focus();

}

}

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

{

status.Text = "";

status.BackColor = Color.DarkCyan;

}

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

{

status.Text = "";

status.BackColor = Color.DarkCyan;

}

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

{

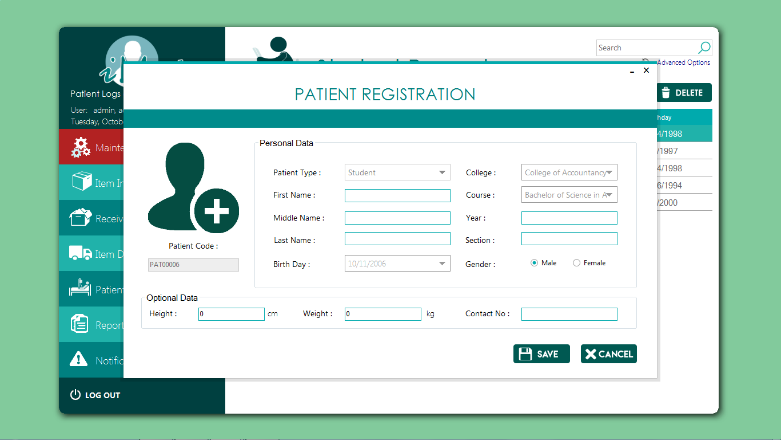
this.Dispose();

}

}

}

**Patient Registration From**

****

using MetroFramework.Controls;

using MetroFramework.Forms;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace PUPiMed

{

public partial class FormAddPatient : MetroForm

{

string strCode, strType, strLName, strFName, strMName, strGender, strDob, strHeight, strWeight, strContactNo, strDept, strCourse, strYear, strSection;

List<String> listDept, listCourse, listCollege;

FormAddPatientLogs log;

private UCStudentTab uCStudentTab;

private UCFacultyTab uCFacultyTab;

private UCAdminTab uCAdminTab;

int intType = 0;

bool new\_ = true;

public FormAddPatient()

{

InitializeComponent();

dtBday.MinDate = DateTime.Now.AddYears(-100);

dtBday.MaxDate = DateTime.Now.AddYears(-10);

}

public FormAddPatient(FormAddPatientLogs log) : this()

{

this.intType = 0;

this.log = log;

this.cbPatientType.SelectedIndex = intType;

loadType();

}

public FormAddPatient(UCStudentTab uCStudentTab) : this()

{

this.uCStudentTab = uCStudentTab;

this.intType = 0;

this.cbPatientType.SelectedIndex = intType;

this.cbPatientType.Enabled = false;

loadType();

}

public FormAddPatient(UCFacultyTab uCFacultyTab) : this()

{

this.uCFacultyTab = uCFacultyTab;

this.intType = 1;

this.cbPatientType.SelectedIndex = intType;

this.cbPatientType.Enabled = false;

loadType();

}

public FormAddPatient(UCAdminTab uCAdminTab) : this()

{

this.uCAdminTab = uCAdminTab;

this.intType = 2;

this.cbPatientType.SelectedIndex = intType;

this.cbPatientType.Enabled = false;

loadType();

}

public FormAddPatient(string code, int type, string first, string middle, string last, string birthday, int gender, string height, string weight, string contact, string college, string course, string year, string section) : this()

{

this.intType = type;

this.cbPatientType.SelectedIndex = intType;

loadType();

this.cbDept.SelectedIndex = listDept.FindIndex(x => x.Equals(college));

if (intType == 0)

{

this.loadCourse();

this.cbCourse.SelectedIndex = listCourse.FindIndex(x => x.Equals(course));

this.txtYear.Text = year;

this.txtSection.Text = section;

}

this.txtCode.Text = code;

this.txtCode.Enabled = false;

this.cbPatientType.Enabled = false;

this.txtFName.Text = first;

this.txtMName.Text = middle;

this.txtLName.Text = last;

this.dtBday.Value = DateTime.Parse(birthday);

rbMale.Checked = gender == 0;

rbFemale.Checked = gender == 1;

this.txtHeight.Text = height;

this.txtWeight.Text = weight;

this.txtContactNo.Text = contact;

new\_ = false;

}

private void loadType()

{

if (intType == 0)

{

cbCourse.Enabled = true;

txtYear.Enabled = true;

txtSection.Enabled = true;

lblDept.Text = "College :";

loadDepartment();

loadCourse();

}

else if (intType == 1)

{

cbCourse.Enabled = false;

txtYear.Enabled = false;

txtSection.Enabled = false;

lblDept.Text = "College :";

loadDepartment();

}

else

{

cbCourse.Enabled = false;

txtYear.Enabled = false;

txtSection.Enabled = false;

lblDept.Text = "Department :";

loadDepartment();

}

}

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

{

getCode();

}

private void getCode()

{

if (new\_)

{

//check if theres an existing code

string prevCode = PUPiMed.Program.getPrevCode("SELECT strPatiCode from tblPatient ORDER by char\_length(strPatiCode) DESC, strPatiCode DESC LIMIT 1;");

if (string.IsNullOrEmpty(prevCode))

{

this.txtCode.BackColor = Color.White;

this.txtCode.Enabled = true;

this.txtCode.Focus();

}

else

{

strCode = PUPiMed.Program.getNextCode(prevCode);

this.txtCode.Text = strCode;

this.txtCode.BackColor = Color.FromArgb(224, 224, 224);

this.txtCode.Enabled = false;

this.cbPatientType.Focus();

}

}

}

private void loadDepartment()

{

string strTable = "";

if (intType == 2)

strTable = "SELECT \* FROM tbldepartment WHERE boolDeptDeleted = FALSE;";

else

strTable = "SELECT \* FROM tblcollege WHERE boolCollDeleted = FALSE;";

if (!Program.fillComboBox(cbDept, strTable, out listDept))

{

cbDept.PromptText = "Nothing Found!";

}

cbDept.SelectedIndex = 0;

}

private void loadCourse()

{

if(cbDept.Items.Count>1)

if (!Program.fillComboBox(cbCourse, "SELECT \* FROM tblCourse WHERE boolCourDeleted = FALSE AND strCourCollCode = '" + listDept[cbDept.SelectedIndex].ToString() +"';", out listCourse))

{

cbCourse.PromptText = "No Course Found!";

}

}

private void metroTextBox3\_Click(object sender, EventArgs e)

{

}

private void metroTextBox4\_Click(object sender, EventArgs e)

{

}

private void metroTextBox2\_Click(object sender, EventArgs e)

{

}

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

{

}

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

{

}

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

{

this.intType = cbPatientType.SelectedIndex;

loadType();

}

private void txtCode\_EnabledChanged(object sender, EventArgs e)

{

MetroTextBox txt = sender as MetroTextBox;

if (txt.Enabled == false)

{

txt.BackColor = Color.FromArgb(224, 224, 224);

}

else

{

txt.BackColor = Color.White;

}

}

private void cbCourse\_EnabledChanged(object sender, EventArgs e)

{

MetroComboBox cb = sender as MetroComboBox;

if (cb.Enabled == false)

{

cb.BackColor = Color.FromArgb(224, 224, 224);

cbCourse.PromptText = "";

}

else

{

cb.BackColor = Color.White;

loadCourse();

}

}

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

{

}

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

{

}

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

{

if (cbPatientType.SelectedIndex == 0)

{

loadCourse();

}

else

{

cbCourse.PromptText = "";

}

}

private bool everythingIsOkay()

{

strCode = txtCode.Text;

strType = cbPatientType.SelectedIndex.ToString();

strDept = listDept[cbDept.SelectedIndex].ToString();

strLName = txtLName.Text;

strFName = txtFName.Text;

strMName = txtMName.Text;

strDob = dtBday.Value.ToString("yyyy-MM-dd");

strGender = rbMale.Checked ? "0" : "1";

strHeight = txtHeight.Text;

strWeight = txtWeight.Text;

strContactNo = txtContactNo.Text;

if (string.IsNullOrEmpty(strCode) || string.IsNullOrEmpty(strLName)||

string.IsNullOrEmpty(strFName)|| string.IsNullOrEmpty(strMName)||

string.IsNullOrEmpty(strType) || string.IsNullOrEmpty(strDept) ||

string.IsNullOrEmpty(strDob) || string.IsNullOrEmpty(strGender))

{

return false;

}

if (intType == 0)

{

strCourse = listCourse[cbCourse.SelectedIndex].ToString();

strYear = txtYear.Text;

strSection = txtSection.Text;

if(string.IsNullOrEmpty(strCourse) || string.IsNullOrEmpty(strYear) || string.IsNullOrEmpty(strSection))

{

return false;

}

}

return true;

}

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

{

MetroTextBox txt = sender as MetroTextBox;

float fltValue;

if (!float.TryParse(txt.Text, out fltValue))

txt.Text = "0";

}

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

{

if (everythingIsOkay())

{

if (string.IsNullOrEmpty(PUPiMed.Program.getNextCode(strCode)))

{

txtCode.Focus();

status.Text = "Invalid code";

status.BackColor = Color.Tomato;

}

else

{

string query;

if (new\_)

query =

"INSERT INTO tblPatient (strPatiCode, intPatiType ,strPatiLastName, strPatiFirstName, strPatiMiddleName, datPatiDOB, intPatiGender, fltPatiHeight, fltPatiWeight, strPatiContact, strPatiAdded) VALUES ('" +

strCode + "','" + intType + "','" + strLName + "','" + strFName + "','" + strMName + "','" + strDob + "','" + strGender + "','" + strHeight + "','" + strWeight + "','" + strContactNo + "', USER());";

else

query = "UPDATE `dbmedicalclinic`.`tblpatient` SET `intPatiType`='" + intType + "', `strPatiLastName`='" + strLName + "', `strPatiFirstName`='" + strFName + "', `strPatiMiddleName`='" + strMName + "', `datPatiDOB`='" + strDob + "', `intPatiGender`='" + strGender + "', `fltPatiHeight`='" + strHeight + "', `fltPatiWeight`='" + strWeight + "', `strPatiContact`='" + strContactNo+"' WHERE `strPatiCode`='"+strCode+"';";

if (!PUPiMed.Program.ExecuteQuery(query))

{

status.Text = "Failed to save patient";

status.BackColor = Color.Tomato;

}

else

{

if (!saveOtherInfo())

{

//tell them something's wrong

status.Text = "Failed to save additional info";

status.BackColor = Color.Tomato;

}

else

{

getCode();

txtCode.Text = strCode;

txtLName.Clear();

txtFName.Clear();

txtMName.Clear();

dtBday.Value = dtBday.MaxDate;

rbMale.Checked = true;

cbPatientType.SelectedIndex = intType;

cbDept.SelectedIndex = 0;

txtYear.Clear();

txtSection.Clear();

txtHeight.Text = "0";

txtWeight.Text = "0";

txtContactNo.Clear();

status.Text = "Saved";

status.BackColor = Color.LimeGreen;

if (log != null)

log.loadPatients(null);

if (uCStudentTab != null)

uCStudentTab.updateTable();

if (uCAdminTab != null)

uCAdminTab.updateTable();

if (uCFacultyTab != null)

uCFacultyTab.updateTable();

if (!new\_)

{

this.Dispose();

}

}

}

}

}

else

{

//tell them it's not :(

status.Text = "Please fill out all the required fields";

status.BackColor = Color.Tomato;

}

}

private bool saveOtherInfo()

{

string query = "";

switch (intType)

{

case 0:

query = "INSERT INTO `dbmedicalclinic`.`tblstudentinfo` (`strStuIPatiCode`, `strStuICourCode`, `strStuIYear`, `strStuISection`) VALUES ('" + strCode + "', '" + strCourse + "', '" + strYear + "', '" + strSection + "');";

break;

case 1:

query = "INSERT INTO `dbmedicalclinic`.`tblfacultyinfo` (`strFacIPatiCode`, `strFacICollCode`) VALUES ('" + strCode + "', '" + strDept + "');";

break;

case 2:

query = "INSERT INTO `dbmedicalclinic`.`tbladmininfo` (`strAdmIPatiCode`, `strAdmIDeptCode`) VALUES ('" + strCode + "', '" + strDept + "');";

break;

}

if (PUPiMed.Program.ExecuteQuery(query))

return true;

return false;

}

private void metroLabel1\_Click(object sender, EventArgs e)

{

}

private void metroComboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void metroLabel9\_Click(object sender, EventArgs e)

{

}

private void metroLabel7\_Click(object sender, EventArgs e)

{

}

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

{

}

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

{

if(this.log!=null)

this.log.loadPatients(null);

if (this.uCAdminTab != null)

this.uCAdminTab.updateTable();

if (this.uCFacultyTab != null)

this.uCFacultyTab.updateTable();

if (this.uCStudentTab != null)

this.uCStudentTab.updateTable();

this.Dispose();

}

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

}

private void height1\_KeyPress(object sender, System.Windows.Forms.KeyPressEventArgs e)

{

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar) && e.KeyChar!= '.';

}

private void cno\_KeyPress(object sender, System.Windows.Forms.KeyPressEventArgs e)

{

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar) && e.KeyChar!='(' && e.KeyChar!=')' && e.KeyChar!='+' && e.KeyChar!='-';

}

private void dtBday\_ValueChanged(object sender, EventArgs e)

{

dtBday.MinDate = DateTime.Now.AddYears(-100);

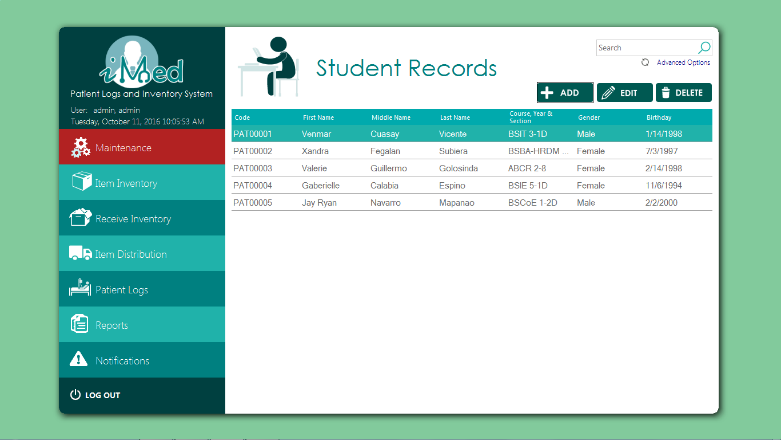
dtBday.MaxDate = DateTime.Now.AddYears(-10);

}

}

}

**Student Records Form**

****

using System;

using System.Data;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using MetroFramework;

namespace PUPiMed

{

public partial class UCStudentTab : UserControl

{

public string strQuery;

FormAddPatient form;

public UCStudentTab()

{

InitializeComponent();

strQuery = "SELECT \* FROM vwstudentlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

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

{

form = new FormAddPatient(this);

form.ShowDialog();

}

public void updateTable()

{

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(strQuery, LoginNew.conn))

{

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

DataTable dt = new DataTable();

try

{

sda.Fill(dt);

grid.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

finally

{

dt.Dispose();

}

}

}

}

catch (Exception exx)

{

MessageBox.Show(exx.Message.ToString());

}

finally

{

LoginNew.conn.Close();

}

}

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

{

strQuery = "SELECT \* FROM vwstudentlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

private void grid\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

getValues();

form.grpOptional.Enabled = false;

form.grpPersonal.Enabled = false;

form.ShowDialog();

}

private void getValues()

{

int gender = 0, type = 0;

string code = "", first = "", middle = "", last = "", birthday = "", height = "", weight = "", contact = "", college = "", course = "", year = "", section = "";

code = grid.CurrentRow.Cells[0].Value.ToString();

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand("SELECT strPatiCode,intPatiType,strPatiLastName,strPatiFirstName,strPatiMiddleName,datPatiDOB,intPatiGender,fltPatiHeight,fltPatiWeight,strPatiContact,strStuICourCode,strStuIYear,strStuISection,strCourCollCode from dbmedicalclinic.tblpatient left join dbmedicalclinic.tblstudentinfo ON strpaticode = strstuiPatiCode left join dbmedicalclinic.tblcourse ON strStuICourCode=strCourCode where strpaticode = '" + code + "';", LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

//Read the data and store them in the list

if (dataReader.Read())

{

code = dataReader.GetString(0);

type = dataReader.GetInt32(1);

last = dataReader.GetString(2);

first = dataReader.GetString(3);

middle = dataReader.GetString(4);

birthday = dataReader.GetString(5);

gender = dataReader.GetInt32(6);

height = dataReader.GetString(7);

weight = dataReader.GetString(8);

contact = dataReader.GetString(9);

course = dataReader.GetString(10);

year = dataReader.GetString(11);

section = dataReader.GetString(12);

college = dataReader.GetString(13);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

form = new FormAddPatient(code, type, first, middle, last, birthday, gender, height, weight, contact, college, course, year, section);

form.FormClosed += Form\_FormClosed;

}

private void Form\_FormClosed(object sender, FormClosedEventArgs e)

{

this.updateTable();

}

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

{

if (grid.CurrentRow != null)

{

var result = MetroMessageBox.Show(this.Parent.Parent, "Are you sure?", "DELETE CONFIRMATION", MessageBoxButtons.YesNo, MessageBoxIcon.Error);

if (result == DialogResult.Yes)

{

string query;

string col1;

if (grid.CurrentRow.Cells[0] != null)

{

col1 = grid.CurrentRow.Cells[0].Value.ToString();

query = "UPDATE tblPatient SET boolPatiDeleted=1, dtmPatiDeleted=NOW(), strPatiDeleted=CURRENT\_USER WHERE strPatiCode = '" + col1 + "';";

if (Program.ExecuteQuery(query))

{

//success

query = "SELECT \* FROM vwstudentlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

}

}

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

strQuery = "SELECT \* FROM vwstudentlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

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

{

if (grid.CurrentRow != null)

{

getValues();

form.ShowDialog();

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

FormPatientSearch form = new FormPatientSearch(this);

strQuery = "SELECT \* FROM vwstudentlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%' ";

form.ShowDialog();

}

}

}

**Faculty Records From**

****

using System;

using System.Data;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using MetroFramework;

namespace PUPiMed

{

public partial class UCFacultyTab : UserControl

{

public string strQuery;

FormAddPatient form;

public UCFacultyTab()

{

InitializeComponent();

strQuery = "SELECT \* FROM vwfacultylogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

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

{

form = new FormAddPatient(this);

form.ShowDialog();

}

public void updateTable()

{

try

{

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(strQuery, LoginNew.conn))

{

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

DataTable dt = new DataTable();

try

{

sda.Fill(dt);

grid.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

finally

{

dt.Dispose();

}

}

}

}

catch (Exception exx)

{

MessageBox.Show(exx.Message.ToString());

}

finally

{

LoginNew.conn.Close();

}

}

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

{

strQuery = "SELECT \* FROM vwfacultylogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

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

{

if (grid.CurrentRow != null)

{

var result = MetroMessageBox.Show(this.Parent.Parent, "Are you sure?", "DELETE CONFIRMATION", MessageBoxButtons.YesNo, MessageBoxIcon.Error);

if (result == DialogResult.Yes)

{

string query;

string col1;

if (grid.CurrentRow.Cells[0] != null)

{

col1 = grid.CurrentRow.Cells[0].Value.ToString();

query = "UPDATE tblPatient SET boolPatiDeleted=1, dtmPatiDeleted=NOW(), strPatiDeleted=CURRENT\_USER WHERE strPatiCode = '" + col1 + "';";

if (Program.ExecuteQuery(query))

{

//success

strQuery = "SELECT \* FROM vwfacultylogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

}

}

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

FormPatientSearch form = new FormPatientSearch(this);

strQuery = "SELECT \* FROM vwfacultylogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%' ";

form.ShowDialog();

}

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

{

if (grid.CurrentRow != null)

{

getValues();

form.ShowDialog();

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void getValues()

{

int gender = 0, type = 0;

string code = "", first = "", middle = "", last = "", birthday = "", height = "", weight = "", contact = "", college = "", course = "", year = "", section = "";

code = grid.CurrentRow.Cells[0].Value.ToString();

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand("SELECT strPatiCode,intPatiType,strPatiLastName,strPatiFirstName,strPatiMiddleName,datPatiDOB,intPatiGender,fltPatiHeight,fltPatiWeight,strPatiContact,strFacICollCode from dbmedicalclinic.tblpatient left join dbmedicalclinic.tblfacultyinfo ON strpaticode = strfaciPatiCode where strpaticode = '" + code + "';", LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

//Read the data and store them in the list

if (dataReader.Read())

{

code = dataReader.GetString(0);

type = dataReader.GetInt32(1);

last = dataReader.GetString(2);

first = dataReader.GetString(3);

middle = dataReader.GetString(4);

birthday = dataReader.GetString(5);

gender = dataReader.GetInt32(6);

height = dataReader.GetString(7);

weight = dataReader.GetString(8);

contact = dataReader.GetString(9);

college = dataReader.GetString(10);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

form = new FormAddPatient(code, type, first, middle, last, birthday, gender, height, weight, contact, college, course, year, section);

form.FormClosed += Form\_FormClosed;

}

private void Form\_FormClosed(object sender, FormClosedEventArgs e)

{

this.updateTable();

}

private void grid\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

getValues();

form.grpOptional.Enabled = false;

form.grpPersonal.Enabled = false;

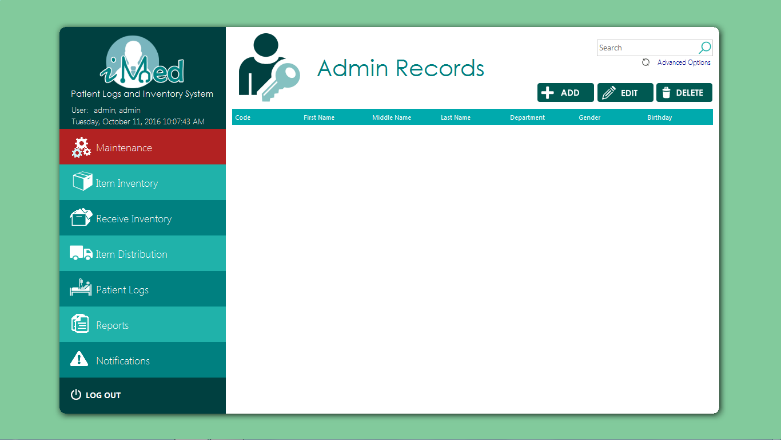
form.ShowDialog();

}

}

}

**Admin Records Form**

****

using System;

using System.Data;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using MetroFramework;

namespace PUPiMed

{

public partial class UCAdminTab : UserControl

{

public string strQuery;

FormAddPatient formpatient;

FormAddPatient form;

public UCAdminTab()

{

InitializeComponent();

strQuery = "SELECT \* FROM vwadminlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

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

{

formpatient = new FormAddPatient(this);

formpatient.ShowDialog();

}

public void updateTable()

{

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(strQuery, LoginNew.conn))

{

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

DataTable dt = new DataTable();

try

{

sda.Fill(dt);

grid.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

finally

{

dt.Dispose();

}

}

}

}

catch (Exception exx)

{

MessageBox.Show(exx.Message.ToString());

}

finally

{

LoginNew.conn.Close();

}

}

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

{

strQuery = "SELECT \* FROM vwadminlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

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

{

if (grid.CurrentRow != null)

{

getValues();

form.ShowDialog();

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void getValues()

{

int gender = 0, type = 0;

string code = "", first = "", middle = "", last = "", birthday = "", height = "", weight = "", contact = "", college = "", course = "", year = "", section = "";

code = grid.CurrentRow.Cells[0].Value.ToString();

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand("SELECT strPatiCode,intPatiType,strPatiLastName,strPatiFirstName,strPatiMiddleName,datPatiDOB,intPatiGender,fltPatiHeight,fltPatiWeight,strPatiContact,strAdmIDeptCode from dbmedicalclinic.tblpatient left join dbmedicalclinic.tbladmininfo ON strpaticode = stradmiPatiCode where strpaticode = '" + code + "';", LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

//Read the data and store them in the list

if (dataReader.Read())

{

code = dataReader.GetString(0);

type = dataReader.GetInt32(1);

last = dataReader.GetString(2);

first = dataReader.GetString(3);

middle = dataReader.GetString(4);

birthday = dataReader.GetString(5);

gender = dataReader.GetInt32(6);

height = dataReader.GetString(7);

weight = dataReader.GetString(8);

contact = dataReader.GetString(9);

college = dataReader.GetString(10);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

form = new FormAddPatient(code, type, first, middle, last, birthday, gender, height, weight, contact, college, course, year, section);

form.FormClosed += Form\_FormClosed;

}

private void Form\_FormClosed(object sender, FormClosedEventArgs e)

{

this.updateTable();

}

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

{

if (grid.CurrentRow != null)

{

var result = MetroMessageBox.Show(this.Parent.Parent, "Are you sure?", "DELETE CONFIRMATION", MessageBoxButtons.YesNo, MessageBoxIcon.Error);

if (result == DialogResult.Yes)

{

string query;

string col1;

if (grid.CurrentRow.Cells[0] != null)

{

col1 = grid.CurrentRow.Cells[0].Value.ToString();

query = "UPDATE tblPatient SET boolPatiDeleted=1, dtmPatiDeleted=NOW(), strPatiDeleted=CURRENT\_USER WHERE strPatiCode = '" + col1 + "';";

if (Program.ExecuteQuery(query))

{

//success

strQuery = "SELECT \* FROM vwadminlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

}

}

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

FormPatientSearch form = new FormPatientSearch(this);

strQuery = "SELECT \* FROM vwadminlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%' ";

form.ShowDialog();

}

private void grid\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

getValues();

form.grpOptional.Enabled = false;

form.grpPersonal.Enabled = false;

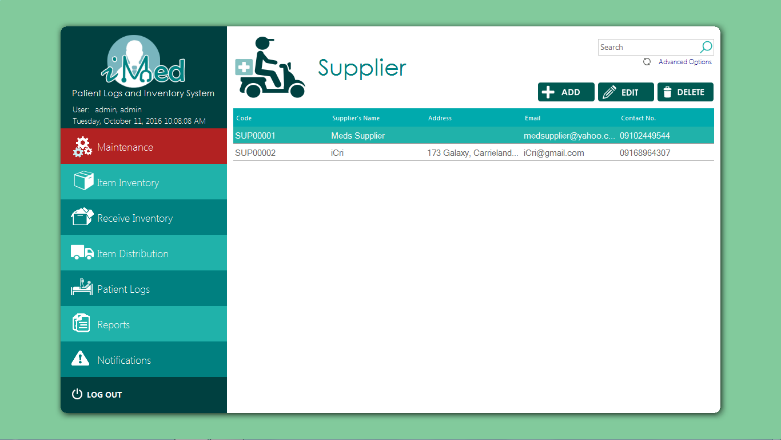
form.ShowDialog();

}

}

}

**Supplier form**

****

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MetroFramework;

using MySql.Data.MySqlClient;

namespace PUPiMed.UserControls

{

public partial class UCSupplier : UserControl

{

FormAddSupplier supplierForm;

public string query;

public UCSupplier()

{

InitializeComponent();

query = "select \* from vwsupplier WHERE `Supplier's Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

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

{

new FormSupplierSearch(this).ShowDialog();

}

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

{

new FormAddSupplier(this,true).ShowDialog();

}

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

{

if (grid.CurrentRow != null)

{

supplierForm = new FormAddSupplier(this, false);

getValues();

supplierForm.ShowDialog();

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void getValues()

{

string code = grid.CurrentRow.Cells[0].Value.ToString();

supplierForm.txtCode.Text = code;

supplierForm.txtName.Text = grid.CurrentRow.Cells[1].Value.ToString();

supplierForm.txtContact.Text = grid.CurrentRow.Cells[3].Value.ToString();

supplierForm.txtEmail.Text = grid.CurrentRow.Cells[4].Value.ToString();

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand("SELECT \* FROM tblAddress a INNER JOIN tblsupplier b ON a.strAddrCode=b.strSuppAddrCode WHERE b.strSuppCode = '" + code + "';", LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

//Read the data and store them in the list

if (dataReader.Read())

{

supplierForm.strACode = dataReader.GetString(0);

supplierForm.txtNo.Text = dataReader.GetString(1);

supplierForm.txtStreet.Text = dataReader.GetString(2);

supplierForm.txtTown.Text = dataReader.GetString(3);

supplierForm.txtCity.Text = dataReader.GetString(4);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

}

public void updateTable()

{

using (LoginNew.conn)

{

using (MySqlCommand cmd = new MySqlCommand(query, LoginNew.conn))

{

cmd.CommandType = CommandType.Text;

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

using (DataTable dt = new DataTable())

{

if (dt != null)

{

sda.Fill(dt);

grid.DataSource = dt;

}

}

}

}

}

}

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

{

if (grid.CurrentRow != null)

{

var result = MetroMessageBox.Show(this.Parent.Parent, "Are you sure?", "DELETE CONFIRMATION", MessageBoxButtons.YesNo, MessageBoxIcon.Error);

if (result == DialogResult.Yes)

{

string strQuery;

string col1;

if (grid.CurrentRow.Cells[0] != null)

{

col1 = grid.CurrentRow.Cells[0].Value.ToString();

strQuery = "UPDATE tblSupplier SET boolSuppDeleted=1, dtmSuppDeleted=NOW(), strSuppDeleted=CURRENT\_USER WHERE strSuppCode = '" + col1 + "';";

if (Program.ExecuteQuery(strQuery))

{

//success

updateTable();

}

else

{

//fail

}

}

}

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

query = "select \* from vwsupplier WHERE `Supplier's Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

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

{

}

private void grid\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (grid.CurrentRow != null)

{

supplierForm = new FormAddSupplier(this, false);

getValues();

supplierForm.supplier.Enabled = false;

supplierForm.ShowDialog();

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

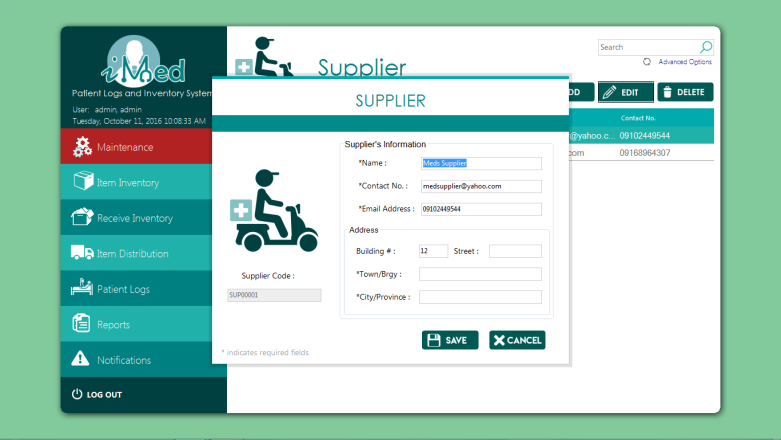
}

}

}

}

**Add Supplier Form**

****

using MetroFramework.Forms;

using System;

using System.Drawing;

using System.Windows.Forms;

using PUPiMed.UserControls;

namespace PUPiMed

{

public partial class FormAddSupplier : MetroForm

{

public string strCode, strName, strACode, strContact, strEmail, strNo, strStreet, strTown, strCity;

private FormReceiveInventory formReceiveInventory;

private UCSupplier uCSupplier;

private bool new\_;

public FormAddSupplier()

{

InitializeComponent();

getCode();

}

public FormAddSupplier(FormReceiveInventory formReceiveInventory) : this()

{

this.formReceiveInventory = formReceiveInventory;

this.new\_ = true;

}

public FormAddSupplier(UCSupplier uCSupplier) : this()

{

this.uCSupplier = uCSupplier;

this.new\_ = true;

}

public FormAddSupplier(UCSupplier uCSupplier, bool new\_) : this(uCSupplier)

{

this.new\_ = new\_;

}

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

{

status.Text = "";

status.BackColor = Color.DarkCyan;

if (string.IsNullOrEmpty(txtCode.Text))

txtCode.ForeColor = Color.White;

}

private void groupBox1\_Enter(object sender, EventArgs e)

{

}

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

{

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);

}

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

{

}

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

{

}

private void getCode()

{

string prevCode = Program.getPrevCode("SELECT strSuppCode from tblSupplier ORDER by char\_length(strSuppCode) DESC, strSuppCode DESC LIMIT 1;");

if (string.IsNullOrEmpty(prevCode))

{

this.txtCode.Enabled = true;

this.txtCode.Focus();

}

else

{

strCode = Program.getNextCode(prevCode);

this.txtCode.Text = strCode;

this.txtCode.Enabled = false;

this.txtName.Focus();

}

}

private bool everythingIsOkay()

{

strCode = txtCode.Text;

strName = txtName.Text;

strContact = txtContact.Text;

strEmail = txtEmail.Text;

strNo = txtNo.Text;

strStreet = txtStreet.Text;

strTown = txtTown.Text;

strCity = txtCity.Text;

if (string.IsNullOrEmpty(strCode)|| string.IsNullOrEmpty(strName) || string.IsNullOrEmpty(strContact) || string.IsNullOrEmpty(strEmail) || string.IsNullOrEmpty(strTown) || string.IsNullOrEmpty(strCity))

{

//check if email is valid

return false;

}

return true;

}

private bool addressIsOkay()

{

//check if there's a prev address code

//default ADDR-000

//increment if there's one

string prevCode = Program.getPrevCode("SELECT strAddrCode from tblAddress ORDER by char\_length(strAddrCode) DESC, strAddrCode DESC LIMIT 1; ");

if (string.IsNullOrEmpty(prevCode))

{

strACode = "ADDR-000";

}

else

{

if (new\_)

strACode = Program.getNextCode(prevCode);

}

return saveAddress();

}

private bool saveAddress()

{

string query;

if (new\_)

{

query = "INSERT INTO tblAddress (`strAddrCode`, `strAddrNum`, `strAddrStreet`, `strAddrTown`, `strAddrCity`, strAddrAdded) VALUES('" + strACode + "','" + strNo + "','" + strStreet + "','" + strTown + "','" + strCity + "', USER());";

}

else

{

query = "UPDATE tblAddress SET strAddrNum='" + strNo + "', strAddrStreet='" + strStreet + "', strAddrTown='" + strTown + "', strAddrCity='" + strCity + "' WHERE strAddrCode='" + strACode + "';";

}

if (!Program.ExecuteQuery(query))

{

status.Text = "Failed to save address";

status.BackColor = Color.Tomato;

return false;

}

return true;

}

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

{

if (everythingIsOkay())

{

if (addressIsOkay())

{

if (string.IsNullOrEmpty(Program.getNextCode(strCode)))

{

txtCode.Focus();

status.Text = "Invalid code";

status.BackColor = Color.Tomato;

}

else

{

string query;

if (new\_)

{

query = "INSERT INTO tblSupplier (`strSuppCode`, `strSuppName`, `strSuppAddrCode`, `strSuppContactNo`, `strSuppEmail`, strSuppAdded) VALUES ('" + strCode + "','" + strName + "','" + strACode + "','" + strContact + "','" + strEmail + "',USER());";

}

else

{

query = "UPDATE tblSupplier SET strSuppCode='" + strCode + "', strSuppName='" + strName + "', strSuppAddrCode='" + strACode + "', strSuppContactNo='" + strContact + "', strSuppEmail='" + strEmail + "' WHERE strSuppCode='" + strCode + "';";

}

//save to db

if (!Program.ExecuteQuery(query))

{

//tell them something's wrong

status.Text = "Failed to save supplier";

status.BackColor = Color.Tomato;

}

else

{

strCode = Program.getNextCode(strCode);

txtCode.Text = strCode;

txtName.Clear();

txtName.Focus();

txtContact.Clear();

txtEmail.Clear();

txtNo.Clear();

txtStreet.Clear();

txtTown.Clear();

txtCity.Clear();

status.Text = "Saved";

status.BackColor = Color.LimeGreen;

if (!new\_)

this.Dispose();

if (this.uCSupplier != null)

this.uCSupplier.updateTable();

if (this.formReceiveInventory != null)

this.formReceiveInventory.loadSupplierComboBox();

}

}

}

}

else

{

//tell them it's not :(

status.Text = "Please fill out all the required fields";

status.BackColor = Color.Tomato;

}

}

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

{

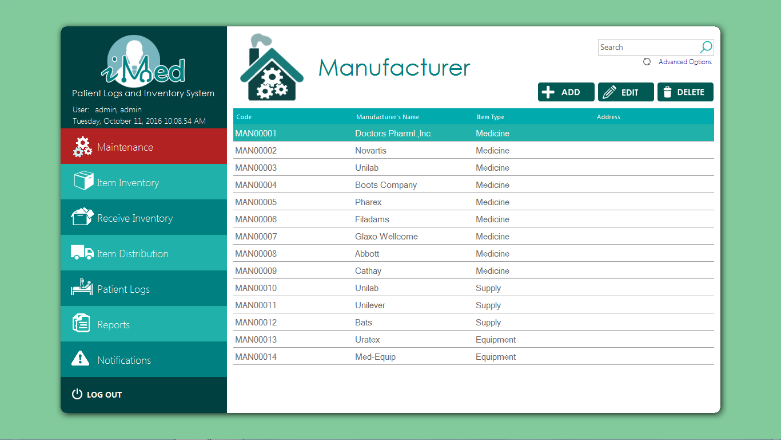
this.Dispose();

}

}

}

**Manufacturer Form**

****

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using PUPiMed.Forms;

using MetroFramework;

using MySql.Data.MySqlClient;

namespace PUPiMed.UserControls

{

public partial class UCManufacturer : UserControl

{

FormAddManufacturer manufacturerForm;

public string query;

public UCManufacturer()

{

InitializeComponent();

}

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

{

new FormManufacturerSearch(this).ShowDialog();

}

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

{

new FormAddManufacturer(this, true).ShowDialog();

}

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

{

if (grid.CurrentRow != null)

{

string code = grid.CurrentRow.Cells[0].Value.ToString();

manufacturerForm = new FormAddManufacturer(this, false);

manufacturerForm.txtCode.Text = code;

manufacturerForm.txtName.Text = grid.CurrentRow.Cells[1].Value.ToString();

manufacturerForm.cbItemType.SelectedIndex = manufacturerForm.cbItemType.Items.IndexOf(grid.CurrentRow.Cells[2].Value.ToString());

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand("SELECT \* FROM tblAddress a INNER JOIN tblManufacturer b ON a.strAddrCode=b.strManuAddrCode WHERE b.strManuCode = '" + code + "';", LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

//Read the data and store them in the list

if (dataReader.Read())

{

manufacturerForm.strACode = dataReader.GetString(0);

manufacturerForm.txtNo.Text = dataReader.GetString(1);

manufacturerForm.txtStreet.Text = dataReader.GetString(2);

manufacturerForm.txtTown.Text = dataReader.GetString(3);

manufacturerForm.txtCity.Text = dataReader.GetString(4);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

manufacturerForm.ShowDialog();

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void updateTable()

{

using (LoginNew.conn)

{

using (MySqlCommand cmd = new MySqlCommand(query, LoginNew.conn))

{

cmd.CommandType = CommandType.Text;

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

using (DataTable dt = new DataTable())

{

if (dt != null)

{

sda.Fill(dt);

grid.DataSource = dt;

}

else

{

}

}

}

}

}

}

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

{

if (grid.CurrentRow != null)

{

var result = MetroMessageBox.Show(this.Parent.Parent, "Are you sure?", "DELETE CONFIRMATION", MessageBoxButtons.YesNo, MessageBoxIcon.Error);

if (result == DialogResult.Yes)

{

string strQuery;

string col1;

if (grid.CurrentRow.Cells[0] != null)

{

col1 = grid.CurrentRow.Cells[0].Value.ToString();

strQuery = "UPDATE tblManufacturer SET boolManuDeleted=1, dtmManuDeleted=NOW(), strManuDeleted=CURRENT\_USER WHERE strManuCode = '" + col1 + "';";

if (Program.ExecuteQuery(strQuery))

{

//success

updateTable();

}

else

{

//fail

}

}

}

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

query = "select \* from vwmanufacturer where `Manufacturer's Name` LIKE '"+txtFilter.Text+"%';";

updateTable();

}

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

{

query = "select \* from vwmanufacturer where `Manufacturer's Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

private void grid\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (grid.CurrentRow != null)

{

string code = grid.CurrentRow.Cells[0].Value.ToString();

manufacturerForm = new FormAddManufacturer(this, false);

manufacturerForm.txtCode.Text = code;

manufacturerForm.txtName.Text = grid.CurrentRow.Cells[1].Value.ToString();

manufacturerForm.cbItemType.SelectedIndex = manufacturerForm.cbItemType.Items.IndexOf(grid.CurrentRow.Cells[2].Value.ToString());

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand("SELECT \* FROM tblAddress a INNER JOIN tblManufacturer b ON a.strAddrCode=b.strManuAddrCode WHERE b.strManuCode = '" + code + "';", LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

//Read the data and store them in the list

if (dataReader.Read())

{

manufacturerForm.strACode = dataReader.GetString(0);

manufacturerForm.txtNo.Text = dataReader.GetString(1);

manufacturerForm.txtStreet.Text = dataReader.GetString(2);

manufacturerForm.txtTown.Text = dataReader.GetString(3);

manufacturerForm.txtCity.Text = dataReader.GetString(4);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

manufacturerForm.manufacturer.Enabled = false;

manufacturerForm.ShowDialog();

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

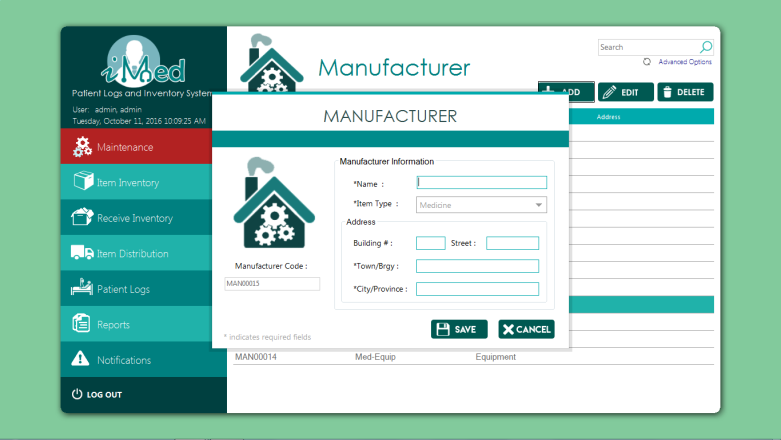
{

}

}

}

**Add Manufacturer Form**

****

using MetroFramework.Forms;

using System;

using System.Drawing;

using PUPiMed.UserControls;

namespace PUPiMed.Forms

{

public partial class FormAddManufacturer : MetroForm

{

public string strCode, strName, strACode, strNo, strStreet, strTown, strCity;

private FormAddItem formAddItem;

private UCManufacturer ucManufacturer;

bool new\_;

public FormAddManufacturer(FormAddItem addItem, int index)

{

InitializeComponent();

this.formAddItem = addItem;

cbItemType.SelectedIndex = index;

cbItemType.Enabled = false;

this.new\_ = true;

getCode();

}

public FormAddManufacturer()

{

InitializeComponent();

getCode();

cbItemType.SelectedIndex = 0;

}

public FormAddManufacturer(UCManufacturer uCManufacturer, bool new\_) : this()

{

this.ucManufacturer = uCManufacturer;

this.new\_ = new\_;

}

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

{

if (this.ucManufacturer != null)

this.ucManufacturer.updateTable();

this.Dispose();

}

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

{

}

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

{

}

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

{

if (everythingIsOkay())

{

if (addressIsOkay())

{

if (string.IsNullOrEmpty(Program.getNextCode(strCode)))

{

txtCode.Focus();

status.Text = "Invalid code";

status.BackColor = Color.Tomato;

}

else

{

string query;

//save to db

if (new\_)

{

query = "INSERT INTO tblmanufacturer (`strManuCode`, `strManuName`, `intManuType`, `strManuAddrCode`, `strManuAdded`) VALUES('" + strCode + "','" + strName + "','" + (cbItemType.SelectedIndex < 1 ? 1 : cbItemType.SelectedIndex + 1) + "','" + strACode + "', USER());";

}

else

{

query = "UPDATE tblManufacturer SET strManuCode='" + strCode + "', intManuType="+ (cbItemType.SelectedIndex < 1 ? 1 : cbItemType.SelectedIndex + 1) + ",strManuName='" + strName + "', strManuAddrCode='" + strACode + "' WHERE strManuCode='" + strCode + "';";

}

//save to db

if (!Program.ExecuteQuery(query))

{

//tell them something's wrong

status.Text = "Failed to save manufacturer";

status.BackColor = Color.Tomato;

}

else

{

strCode = Program.getNextCode(strCode);

txtCode.Text = strCode;

txtName.Clear();

txtName.Focus();

txtNo.Clear();

txtStreet.Clear();

txtTown.Clear();

txtCity.Clear();

cbItemType.SelectedIndex = 0;

status.Text = "Saved";

status.BackColor = Color.LimeGreen;

if (formAddItem != null)

formAddItem.loadManufacturer();

if (!new\_)

{

this.ucManufacturer.updateTable();

this.Dispose();

}

}

}

}

}

else

{

//tell them it's not :(

status.Text = "Please fill out all the required fields";

status.BackColor = Color.Tomato;

}

}

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

{

status.Text = "";

status.BackColor = Color.DarkCyan;

}

private bool everythingIsOkay()

{

strCode = txtCode.Text;

strName = txtName.Text;

strNo = txtNo.Text;

strStreet = txtStreet.Text;

strTown = txtTown.Text;

strCity = txtCity.Text;

if (string.IsNullOrEmpty(strCode) || string.IsNullOrEmpty(strName) || string.IsNullOrEmpty(strTown) || string.IsNullOrEmpty(strCity))

{

return false;

}

return true;

}

private bool addressIsOkay()

{

//check if there's a prev address code

//default ADDR-000

//increment if there's one

string prevCode = Program.getPrevCode("SELECT strAddrCode from tblAddress ORDER by char\_length(strAddrCode) DESC, strAddrCode DESC LIMIT 1;");

if (string.IsNullOrEmpty(prevCode))

{

strACode = "ADDR-000";

}

else

{

if (new\_)

strACode = Program.getNextCode(prevCode);

}

return saveAddress();

}

private bool saveAddress()

{

string query;

if (new\_)

{

query = "INSERT INTO tblAddress (`strAddrCode`, `strAddrNum`, `strAddrStreet`, `strAddrTown`, `strAddrCity`, strAddrAdded) VALUES('" + strACode + "','" + strNo + "','" + strStreet + "','" + strTown + "','" + strCity + "', USER());";

}

else

{

query = "UPDATE tblAddress SET strAddrNum='" + strNo + "', strAddrStreet='" + strStreet + "', strAddrTown='" + strTown + "', strAddrCity='" + strCity + "' WHERE strAddrCode='" + strACode + "';";

}

if (!Program.ExecuteQuery(query))

{

status.Text = "Failed to save address";

status.BackColor = Color.Tomato;

return false;

}

return true;

}

private void getCode()

{

string prevCode = Program.getPrevCode("SELECT strManuCode from tblManufacturer ORDER by char\_length(strManuCode) DESC, strManuCode DESC LIMIT 1;");

if (string.IsNullOrEmpty(prevCode))

{

this.txtCode.Enabled = true;

this.txtCode.Focus();

}

else

{

strCode = Program.getNextCode(prevCode);

this.txtCode.Text = strCode;

this.txtCode.Enabled = false;

this.txtName.Focus();

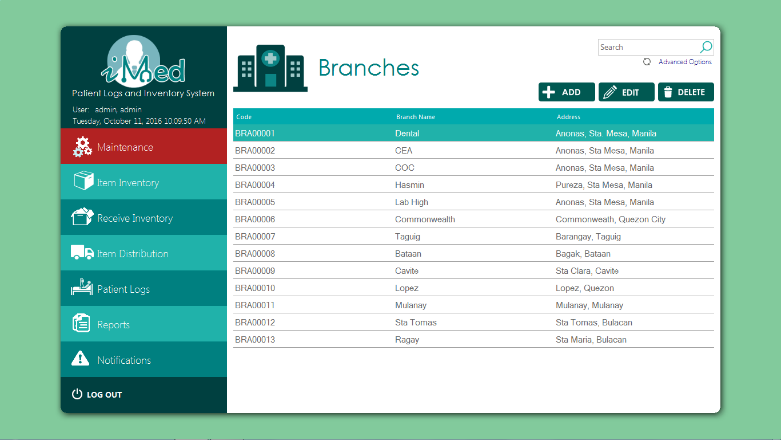
}

}

}

}

**Branches Form**

****

using System;

using System.Data;

using System.Windows.Forms;

using PUPiMed.Forms;

using MetroFramework;

using MySql.Data.MySqlClient;

namespace PUPiMed.UserControls

{

public partial class UCBranches : UserControl

{

FormAddBranches branchesForm;

public string query;

public UCBranches()

{

InitializeComponent();

query = "select \* from vwbranches WHERE `Branch Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

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

{

new FormBranchSearch(this).ShowDialog();

}

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

{

new Forms.FormAddBranches(this,true).ShowDialog();

}

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

{

if (grid.CurrentRow != null)

{

string code = grid.CurrentRow.Cells[0].Value.ToString();

branchesForm = new FormAddBranches(this,false);

branchesForm.txtCode.Text = code;

branchesForm.txtName.Text = grid.CurrentRow.Cells[1].Value.ToString();

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand("SELECT \* FROM tblAddress a INNER JOIN tblBranch b ON a.strAddrCode=b.strBranAddrCode WHERE b.strBranCode = '" + code + "';", LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

//Read the data and store them in the list

if (dataReader.Read())

{

branchesForm.strACode = dataReader.GetString(0);

branchesForm.txtNo.Text = dataReader.GetString(1);

branchesForm.txtStreet.Text = dataReader.GetString(2);

branchesForm.txtTown.Text = dataReader.GetString(3);

branchesForm.txtCity.Text = dataReader.GetString(4);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

branchesForm.ShowDialog();

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public void updateTable()

{

using (LoginNew.conn)

{

using (MySqlCommand cmd = new MySqlCommand(query, LoginNew.conn))

{

cmd.CommandType = CommandType.Text;

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

using (DataTable dt = new DataTable())

{

if (dt != null)

{

sda.Fill(dt);

grid.DataSource = dt;

}

}

}

}

}

}

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

{

query = "select \* from vwbranches WHERE `Branch Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

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

{

if (grid.CurrentRow != null)

{

var result = MetroMessageBox.Show(this.Parent.Parent, "Are you sure?", "DELETE CONFIRMATION", MessageBoxButtons.YesNo, MessageBoxIcon.Error);

if (result == DialogResult.Yes)

{

string strQuery;

string col1;

if (grid.CurrentRow.Cells[0] != null)

{

col1 = grid.CurrentRow.Cells[0].Value.ToString();

strQuery = "UPDATE tblBranch SET boolBranDeleted=1, dtmBranDeleted=NOW(), strBranDeleted=CURRENT\_USER WHERE strBranCode = '" + col1 + "';";

if (Program.ExecuteQuery(strQuery))

{

//success

updateTable();

}

}

}

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void grid\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (grid.CurrentRow != null)

{

string code = grid.CurrentRow.Cells[0].Value.ToString();

branchesForm = new FormAddBranches(this, false);

branchesForm.txtCode.Text = code;

branchesForm.txtName.Text = grid.CurrentRow.Cells[1].Value.ToString();

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand("SELECT \* FROM tblAddress a INNER JOIN tblBranch b ON a.strAddrCode=b.strBranAddrCode WHERE b.strBranCode = '" + code + "';", LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

//Read the data and store them in the list

if (dataReader.Read())

{

branchesForm.strACode = dataReader.GetString(0);

branchesForm.txtNo.Text = dataReader.GetString(1);

branchesForm.txtStreet.Text = dataReader.GetString(2);

branchesForm.txtTown.Text = dataReader.GetString(3);

branchesForm.txtCity.Text = dataReader.GetString(4);

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

branchesForm.branch.Enabled = false;

branchesForm.ShowDialog();

}

else

{

MetroMessageBox.Show(this.Parent.Parent, "No results found.", "EMPTY GRID", MessageBoxButtons.OK, MessageBoxIcon.Error);

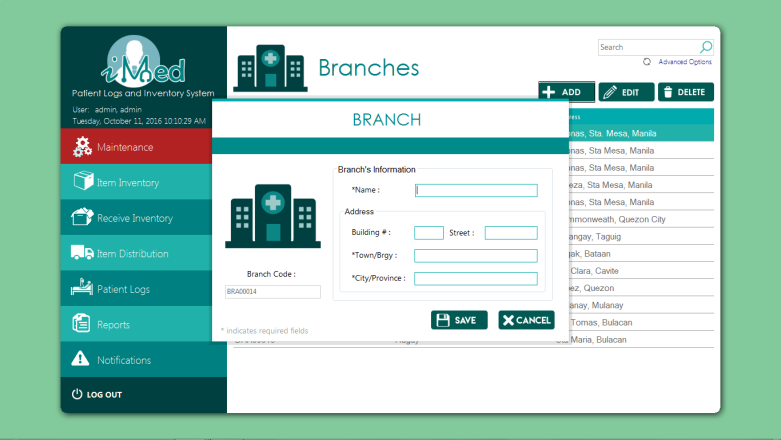
}

}

}

}

**Add Branches**

****

namespace PUPiMed.Forms

{

partial class FormAddBranches

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(FormAddBranches));

this.title = new System.Windows.Forms.Label();

this.status = new MetroFramework.Controls.MetroTile();

this.suppcode = new MetroFramework.Controls.MetroLabel();

this.txtCode = new MetroFramework.Controls.MetroTextBox();

this.branch = new System.Windows.Forms.GroupBox();

this.groupBox1 = new System.Windows.Forms.GroupBox();

this.txtNo = new MetroFramework.Controls.MetroTextBox();

this.metroLabel5 = new MetroFramework.Controls.MetroLabel();

this.txtTown = new MetroFramework.Controls.MetroTextBox();

this.txtCity = new MetroFramework.Controls.MetroTextBox();

this.metroLabel4 = new MetroFramework.Controls.MetroLabel();

this.metroLabel3 = new MetroFramework.Controls.MetroLabel();

this.metroLabel2 = new MetroFramework.Controls.MetroLabel();

this.txtStreet = new MetroFramework.Controls.MetroTextBox();

this.txtName = new MetroFramework.Controls.MetroTextBox();

this.metroLabel1 = new MetroFramework.Controls.MetroLabel();

this.metroLabel7 = new MetroFramework.Controls.MetroLabel();

this.pictureBox12 = new System.Windows.Forms.PictureBox();

this.btnCansel = new System.Windows.Forms.Button();

this.btnSeyb = new System.Windows.Forms.Button();

this.branch.SuspendLayout();

this.groupBox1.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox12)).BeginInit();

this.SuspendLayout();

//

// title

//

this.title.Font = new System.Drawing.Font("Century Gothic", 21.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.title.ForeColor = System.Drawing.Color.Teal;

this.title.Location = new System.Drawing.Point(234, 6);

this.title.Name = "title";

this.title.Size = new System.Drawing.Size(146, 59);

this.title.TabIndex = 146;

this.title.Text = "BRANCH";

this.title.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

//

// status

//

this.status.ActiveControl = null;

this.status.BackColor = System.Drawing.Color.DarkCyan;

this.status.Location = new System.Drawing.Point(-2, 68);

this.status.Name = "status";

this.status.Size = new System.Drawing.Size(632, 29);

this.status.TabIndex = 148;

this.status.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

this.status.UseCustomBackColor = true;

this.status.UseSelectable = true;

//

// suppcode

//

this.suppcode.AccessibleName = "suppcode";

this.suppcode.Anchor = System.Windows.Forms.AnchorStyles.None;

this.suppcode.AutoSize = true;

this.suppcode.FontWeight = MetroFramework.MetroLabelWeight.Regular;

this.suppcode.Location = new System.Drawing.Point(57, 296);

this.suppcode.Name = "suppcode";

this.suppcode.Size = new System.Drawing.Size(94, 19);

this.suppcode.Style = MetroFramework.MetroColorStyle.Teal;

this.suppcode.TabIndex = 152;

this.suppcode.Text = "Branch Code :";

//

// txtCode

//

this.txtCode.AccessibleName = "Eqcode";

this.txtCode.Anchor = System.Windows.Forms.AnchorStyles.None;

this.txtCode.Lines = new string[0];

this.txtCode.Location = new System.Drawing.Point(23, 326);

this.txtCode.MaxLength = 15;

this.txtCode.Name = "txtCode";

this.txtCode.PasswordChar = '\0';

this.txtCode.ScrollBars = System.Windows.Forms.ScrollBars.None;

this.txtCode.SelectedText = "";

this.txtCode.Size = new System.Drawing.Size(167, 23);

this.txtCode.Style = MetroFramework.MetroColorStyle.Teal;

this.txtCode.TabIndex = 1;

this.txtCode.UseCustomBackColor = true;

this.txtCode.UseSelectable = true;

this.txtCode.Click += new System.EventHandler(this.txtCode\_Click);

//

// branch

//

this.branch.Controls.Add(this.groupBox1);

this.branch.Controls.Add(this.txtName);

this.branch.Controls.Add(this.metroLabel1);

this.branch.Font = new System.Drawing.Font("Microsoft Sans Serif", 11.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.branch.Location = new System.Drawing.Point(211, 114);

this.branch.Name = "branch";

this.branch.Size = new System.Drawing.Size(385, 240);

this.branch.TabIndex = 2;

this.branch.TabStop = false;

this.branch.Text = "Branch\'s Information";

//

// groupBox1

//

this.groupBox1.Controls.Add(this.txtNo);

this.groupBox1.Controls.Add(this.metroLabel5);

this.groupBox1.Controls.Add(this.txtTown);

this.groupBox1.Controls.Add(this.txtCity);

this.groupBox1.Controls.Add(this.metroLabel4);

this.groupBox1.Controls.Add(this.metroLabel3);

this.groupBox1.Controls.Add(this.metroLabel2);

this.groupBox1.Controls.Add(this.txtStreet);

this.groupBox1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.groupBox1.Location = new System.Drawing.Point(12, 75);

this.groupBox1.Name = "groupBox1";

this.groupBox1.Size = new System.Drawing.Size(360, 150);

this.groupBox1.TabIndex = 3;

this.groupBox1.TabStop = false;

this.groupBox1.Text = "Address";

//

// txtNo

//

this.txtNo.AccessibleName = "";

this.txtNo.BackColor = System.Drawing.Color.White;

this.txtNo.Lines = new string[0];

this.txtNo.Location = new System.Drawing.Point(131, 34);

this.txtNo.MaxLength = 50;

this.txtNo.Name = "txtNo";

this.txtNo.PasswordChar = '\0';

this.txtNo.ScrollBars = System.Windows.Forms.ScrollBars.None;

this.txtNo.SelectedText = "";

this.txtNo.Size = new System.Drawing.Size(51, 23);

this.txtNo.Style = MetroFramework.MetroColorStyle.Teal;

this.txtNo.TabIndex = 3;

this.txtNo.UseCustomBackColor = true;

this.txtNo.UseSelectable = true;

this.txtNo.UseStyleColors = true;

this.txtNo.Click += new System.EventHandler(this.txtCode\_Click);

//

// metroLabel5

//

this.metroLabel5.AccessibleName = "lbltype";

this.metroLabel5.BackColor = System.Drawing.Color.Transparent;

this.metroLabel5.FontWeight = MetroFramework.MetroLabelWeight.Regular;

this.metroLabel5.Location = new System.Drawing.Point(17, 34);

this.metroLabel5.Name = "metroLabel5";

this.metroLabel5.Size = new System.Drawing.Size(94, 19);

this.metroLabel5.TabIndex = 140;

this.metroLabel5.Text = "Building # :";

this.metroLabel5.UseCustomBackColor = true;

this.metroLabel5.UseCustomForeColor = true;

//

// txtTown

//

this.txtTown.AccessibleName = "";

this.txtTown.BackColor = System.Drawing.Color.White;

this.txtTown.Lines = new string[0];

this.txtTown.Location = new System.Drawing.Point(131, 74);

this.txtTown.MaxLength = 50;

this.txtTown.Name = "txtTown";

this.txtTown.PasswordChar = '\0';

this.txtTown.ScrollBars = System.Windows.Forms.ScrollBars.None;

this.txtTown.SelectedText = "";

this.txtTown.Size = new System.Drawing.Size(215, 23);

this.txtTown.Style = MetroFramework.MetroColorStyle.Teal;

this.txtTown.TabIndex = 5;

this.txtTown.UseCustomBackColor = true;

this.txtTown.UseSelectable = true;

this.txtTown.UseStyleColors = true;

this.txtTown.Click += new System.EventHandler(this.txtCode\_Click);

//

// txtCity

//

this.txtCity.AccessibleName = "";

this.txtCity.BackColor = System.Drawing.Color.White;

this.txtCity.Lines = new string[0];

this.txtCity.Location = new System.Drawing.Point(131, 114);

this.txtCity.MaxLength = 50;

this.txtCity.Name = "txtCity";

this.txtCity.PasswordChar = '\0';

this.txtCity.ScrollBars = System.Windows.Forms.ScrollBars.None;

this.txtCity.SelectedText = "";

this.txtCity.Size = new System.Drawing.Size(215, 23);

this.txtCity.Style = MetroFramework.MetroColorStyle.Teal;

this.txtCity.TabIndex = 6;

this.txtCity.UseCustomBackColor = true;

this.txtCity.UseSelectable = true;

this.txtCity.UseStyleColors = true;

this.txtCity.Click += new System.EventHandler(this.txtCode\_Click);

//

// metroLabel4

//

this.metroLabel4.AccessibleName = "lbltype";

this.metroLabel4.BackColor = System.Drawing.Color.Transparent;

this.metroLabel4.FontWeight = MetroFramework.MetroLabelWeight.Regular;

this.metroLabel4.Location = new System.Drawing.Point(17, 114);

this.metroLabel4.Name = "metroLabel4";

this.metroLabel4.Size = new System.Drawing.Size(105, 19);

this.metroLabel4.TabIndex = 137;

this.metroLabel4.Text = "\*City/Province :";

this.metroLabel4.UseCustomBackColor = true;

this.metroLabel4.UseCustomForeColor = true;

//

// metroLabel3

//

this.metroLabel3.AccessibleName = "lbltype";

this.metroLabel3.BackColor = System.Drawing.Color.Transparent;

this.metroLabel3.FontWeight = MetroFramework.MetroLabelWeight.Regular;

this.metroLabel3.Location = new System.Drawing.Point(17, 74);

this.metroLabel3.Name = "metroLabel3";

this.metroLabel3.Size = new System.Drawing.Size(94, 19);

this.metroLabel3.TabIndex = 136;

this.metroLabel3.Text = "\*Town/Brgy :";

this.metroLabel3.UseCustomBackColor = true;

this.metroLabel3.UseCustomForeColor = true;

//

// metroLabel2

//

this.metroLabel2.AccessibleName = "lbltype";

this.metroLabel2.BackColor = System.Drawing.Color.Transparent;

this.metroLabel2.FontWeight = MetroFramework.MetroLabelWeight.Regular;

this.metroLabel2.Location = new System.Drawing.Point(188, 35);

this.metroLabel2.Name = "metroLabel2";

this.metroLabel2.Size = new System.Drawing.Size(60, 19);

this.metroLabel2.TabIndex = 135;

this.metroLabel2.Text = "Street :";

this.metroLabel2.UseCustomBackColor = true;

this.metroLabel2.UseCustomForeColor = true;

//

// txtStreet

//

this.txtStreet.AccessibleName = "";

this.txtStreet.BackColor = System.Drawing.Color.White;

this.txtStreet.Lines = new string[0];

this.txtStreet.Location = new System.Drawing.Point(254, 34);

this.txtStreet.MaxLength = 50;

this.txtStreet.Name = "txtStreet";

this.txtStreet.PasswordChar = '\0';

this.txtStreet.ScrollBars = System.Windows.Forms.ScrollBars.None;

this.txtStreet.SelectedText = "";

this.txtStreet.Size = new System.Drawing.Size(92, 23);

this.txtStreet.Style = MetroFramework.MetroColorStyle.Teal;

this.txtStreet.TabIndex = 4;

this.txtStreet.UseCustomBackColor = true;

this.txtStreet.UseSelectable = true;

this.txtStreet.UseStyleColors = true;

this.txtStreet.Click += new System.EventHandler(this.txtCode\_Click);

//

// txtName

//

this.txtName.AccessibleName = "";

this.txtName.Lines = new string[0];

this.txtName.Location = new System.Drawing.Point(144, 35);

this.txtName.MaxLength = 100;

this.txtName.Name = "txtName";

this.txtName.PasswordChar = '\0';

this.txtName.ScrollBars = System.Windows.Forms.ScrollBars.None;

this.txtName.SelectedText = "";

this.txtName.Size = new System.Drawing.Size(214, 23);

this.txtName.Style = MetroFramework.MetroColorStyle.Teal;

this.txtName.TabIndex = 2;

this.txtName.UseCustomBackColor = true;

this.txtName.UseSelectable = true;

this.txtName.UseStyleColors = true;

this.txtName.Click += new System.EventHandler(this.txtCode\_Click);

//

// metroLabel1

//

this.metroLabel1.AccessibleName = "lblmed";

this.metroLabel1.AutoSize = true;

this.metroLabel1.BackColor = System.Drawing.Color.Transparent;

this.metroLabel1.FontWeight = MetroFramework.MetroLabelWeight.Regular;

this.metroLabel1.Location = new System.Drawing.Point(29, 35);

this.metroLabel1.Name = "metroLabel1";

this.metroLabel1.Size = new System.Drawing.Size(58, 19);

this.metroLabel1.TabIndex = 124;

this.metroLabel1.Text = "\*Name :";

this.metroLabel1.UseCustomBackColor = true;

//

// metroLabel7

//

this.metroLabel7.AccessibleName = "suppcode";

this.metroLabel7.AutoSize = true;

this.metroLabel7.FontWeight = MetroFramework.MetroLabelWeight.Regular;

this.metroLabel7.ForeColor = System.Drawing.SystemColors.ControlDark;

this.metroLabel7.Location = new System.Drawing.Point(11, 395);

this.metroLabel7.Name = "metroLabel7";

this.metroLabel7.Size = new System.Drawing.Size(162, 19);

this.metroLabel7.Style = MetroFramework.MetroColorStyle.Teal;

this.metroLabel7.TabIndex = 161;

this.metroLabel7.Text = "\* indicates required fields";

this.metroLabel7.UseCustomForeColor = true;

//

// pictureBox12

//

this.pictureBox12.Anchor = System.Windows.Forms.AnchorStyles.None;

this.pictureBox12.Image = global::PUPiMed.Properties.Resources.branchesds\_\_3\_;

this.pictureBox12.Location = new System.Drawing.Point(23, 121);

this.pictureBox12.Name = "pictureBox12";

this.pictureBox12.Size = new System.Drawing.Size(167, 163);

this.pictureBox12.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;

this.pictureBox12.TabIndex = 149;

this.pictureBox12.TabStop = false;

//

// btnCansel

//

this.btnCansel.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Left)));

this.btnCansel.BackColor = System.Drawing.Color.Transparent;

this.btnCansel.BackgroundImage = global::PUPiMed.Properties.Resources.cancel;

this.btnCansel.BackgroundImageLayout = System.Windows.Forms.ImageLayout.Zoom;

this.btnCansel.Cursor = System.Windows.Forms.Cursors.Hand;

this.btnCansel.FlatAppearance.BorderSize = 0;

this.btnCansel.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.btnCansel.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.btnCansel.Location = new System.Drawing.Point(501, 368);

this.btnCansel.Name = "btnCansel";

this.btnCansel.Size = new System.Drawing.Size(98, 37);

this.btnCansel.TabIndex = 190;

this.btnCansel.UseVisualStyleBackColor = false;

this.btnCansel.Click += new System.EventHandler(this.btnCancel\_Click);

//

// btnSeyb

//

this.btnSeyb.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Left)));

this.btnSeyb.BackColor = System.Drawing.Color.Transparent;

this.btnSeyb.BackgroundImage = global::PUPiMed.Properties.Resources.save2;

this.btnSeyb.BackgroundImageLayout = System.Windows.Forms.ImageLayout.Zoom;

this.btnSeyb.Cursor = System.Windows.Forms.Cursors.Hand;

this.btnSeyb.FlatAppearance.BorderSize = 0;

this.btnSeyb.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.btnSeyb.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.btnSeyb.Location = new System.Drawing.Point(383, 368);

this.btnSeyb.Name = "btnSeyb";

this.btnSeyb.Size = new System.Drawing.Size(99, 37);

this.btnSeyb.TabIndex = 189;

this.btnSeyb.UseVisualStyleBackColor = false;

this.btnSeyb.Click += new System.EventHandler(this.btnAdd\_Click);

//

// FormAddBranches

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(624, 424);

this.ControlBox = false;

this.Controls.Add(this.btnCansel);

this.Controls.Add(this.btnSeyb);

this.Controls.Add(this.metroLabel7);

this.Controls.Add(this.branch);

this.Controls.Add(this.suppcode);

this.Controls.Add(this.txtCode);

this.Controls.Add(this.pictureBox12);

this.Controls.Add(this.status);

this.Controls.Add(this.title);

this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));

this.Name = "FormAddBranches";

this.Resizable = false;

this.Style = MetroFramework.MetroColorStyle.Teal;

this.branch.ResumeLayout(false);

this.branch.PerformLayout();

this.groupBox1.ResumeLayout(false);

((System.ComponentModel.ISupportInitialize)(this.pictureBox12)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

public MetroFramework.Controls.MetroTextBox txtCode;

public MetroFramework.Controls.MetroTextBox txtNo;

public MetroFramework.Controls.MetroTextBox txtTown;

public MetroFramework.Controls.MetroTextBox txtCity;

public MetroFramework.Controls.MetroTextBox txtStreet;

public MetroFramework.Controls.MetroTextBox txtName;

public System.Windows.Forms.Label title;

public MetroFramework.Controls.MetroTile status;

public System.Windows.Forms.PictureBox pictureBox12;

public MetroFramework.Controls.MetroLabel suppcode;

public System.Windows.Forms.GroupBox branch;

public System.Windows.Forms.GroupBox groupBox1;

public MetroFramework.Controls.MetroLabel metroLabel5;

public MetroFramework.Controls.MetroLabel metroLabel4;

public MetroFramework.Controls.MetroLabel metroLabel3;

public MetroFramework.Controls.MetroLabel metroLabel2;

public MetroFramework.Controls.MetroLabel metroLabel1;

public MetroFramework.Controls.MetroLabel metroLabel7;

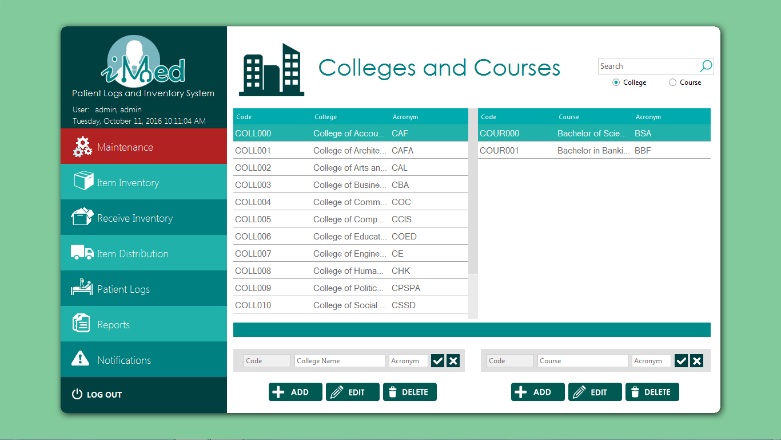
private System.Windows.Forms.Button btnCansel;

private System.Windows.Forms.Button btnSeyb;

}

}

**Colleges and Courses Form**

****

using MySql.Data.MySqlClient;

using System;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

namespace PUPiMed

{

public partial class UCCollege : UserControl

{

string strCode1, strName1, strAcronym1, strCode2, strName2, strAcronym2, query1, query2, query;

bool newRecord1, newRecord2;

public UCCollege()

{

InitializeComponent();

query1 = "SELECT strCollCode AS 'Code', strCollName AS 'College', IFNULL(strCollShort,'') AS 'Acronym' FROM tblCollege WHERE boolCollDeleted=FALSE;";

updateTable1();

}

#region College

public void updateTable1()

{

using(LoginNew.conn)

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(query1, LoginNew.conn))

{

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

using (DataTable dt = new DataTable())

{

if (dt != null)

{

sda.Fill(dt);

grid.DataSource = dt;

}

}

}

}

}

}

private void grid\_CurrentCellChanged(object sender, EventArgs e)

{

if (grid.CurrentRow != null)

{

if (rb2.Checked)

query2 = "SELECT strCourCode AS 'Code', strCourName AS 'Course', IFNULL(strCourShort,'') AS 'Acronym' FROM tblCourse WHERE boolCourDeleted=FALSE AND strCourName LIKE '" + txtFilter.Text + "%' AND strCourCollCode LIKE '" + grid.CurrentRow.Cells[0].Value.ToString() + "';";

else

query2 = "SELECT strCourCode AS 'Code', strCourName AS 'Course', IFNULL(strCourShort,'') AS 'Acronym' FROM tblCourse WHERE boolCourDeleted=FALSE AND strCourCollCode LIKE '" + grid.CurrentRow.Cells[0].Value.ToString() + "';";

updateTable2();

txtCode2.Enabled = false;

txtName2.Enabled = true;

txtAcronym2.Enabled = true;

btnSave2.Enabled = true;

btnEdit2.Enabled = true;

btnDelete2.Enabled = true;

btnAdd2.Enabled = true;

btnClear2.Enabled = true;

}

else

{

txtCode2.Enabled = false;

txtName2.Enabled = false;

txtAcronym2.Enabled = false;

btnSave2.Enabled = false;

btnEdit2.Enabled = false;

btnDelete2.Enabled = false;

btnAdd2.Enabled = false;

btnClear2.Enabled = false;

}

}

private void btnAdd1\_Click(object sender, EventArgs e)

{

newRecord1 = true;

clearContents1();

getCode1();

txtName1.Focus();

}

private bool everythingIsOkay1()

{

getCode1();

strCode1 = txtCode1.Text;

strAcronym1 = txtAcronym1.Text;

strName1 = txtName1.Text;

if (string.IsNullOrEmpty(strCode1) || string.IsNullOrEmpty(strAcronym1) || string.IsNullOrEmpty(strName1))

return false;

return true;

}

private void clearContents1()

{

txtAcronym1.Clear();

txtName1.Clear();

}

private void getCode1()

{

strCode1 = Program.getPrevCode("SELECT strCollCode from tblCollege ORDER by char\_length(strCollCode) DESC,strCollCode DESC LIMIT 1;");

if (string.IsNullOrEmpty(strCode1))

{

txtCode1.Enabled = true;

}

else

{

strCode1 = Program.getNextCode(strCode1);

txtCode1.Text = strCode1;

txtCode1.Enabled = false;

}

txtCode1.BackColor = txtCode1.Enabled == true ? Color.White : Color.FromName("Control");

}

private void grid\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (grid.CurrentRow != null)

{

txtCode1.Text = grid.CurrentRow.Cells[0].Value.ToString();

txtName1.Text = grid.CurrentRow.Cells[1].Value.ToString();

txtAcronym1.Text = grid.CurrentRow.Cells[2].Value.ToString();

}

}

private void btnDelete1\_Click(object sender, EventArgs e)

{

if (grid.CurrentRow != null)

{

if (MetroFramework.MetroMessageBox.Show(this.Parent.Parent, "Are you sure?", "DELETE CONFIRMATION", MessageBoxButtons.YesNo, MessageBoxIcon.Error) == DialogResult.Yes)

{

if (Program.ExecuteQuery("UPDATE tblCollege SET boolCollDeleted = TRUE, dtmCollDeleted=NOW(), strCollDeleted=CURRENT\_USER WHERE strCollCode = '" + grid.CurrentRow.Cells[0].Value.ToString() + "';"))

{

query1 = "SELECT strCollCode AS 'Code', strCollName AS 'College', IFNULL(strCollShort,'') AS 'Acronym' FROM tblCollege WHERE boolCollDeleted=FALSE;";

updateTable1();

}

}

}

}

private void btnEdit1\_Click(object sender, EventArgs e)

{

if (grid.CurrentRow != null)

{

txtCode1.Text = grid.CurrentRow.Cells[0].Value.ToString();

txtName1.Text = grid.CurrentRow.Cells[1].Value.ToString();

txtAcronym1.Text = grid.CurrentRow.Cells[2].Value.ToString();

newRecord1 = false;

}

}

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

{

clearContents1();

}

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

{

if (everythingIsOkay1())

{

if (newRecord1)

query = "INSERT INTO `dbmedicalclinic`.`tblcollege` (`strCollCode`, `strCollName`, `strCollShort`, `strCollAdded`) VALUES ('" + strCode1 + "', '" + strName1 + "', '" + strAcronym1 + "', USER());";

else

query = "UPDATE `dbmedicalclinic`.`tblcollege` SET `strCollName`='"+strName1+"', `strCollShort`='"+strAcronym1+"' WHERE `strCollCode`='"+strCode1+"';";

if (!Program.ExecuteQuery(query))

{

if (newRecord1)

error("Failed to save college");

else

error("Failed to update college");

}

else

{

clearContents1();

status.Text = "Saved";

status.BackColor = Color.LimeGreen;

updateTable1();

getCode1();

}

}

else

{

error("Please fill out empty fields");

}

}

#endregion

#region Course

public void updateTable2()

{

using (LoginNew.conn)

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(query2, LoginNew.conn))

{

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

using (DataTable dt = new DataTable())

{

if (dt != null)

{

sda.Fill(dt);

grid2.DataSource = dt;

}

}

}

}

}

}

private void getCode2()

{

strCode2 = Program.getPrevCode("SELECT strCourCode from tblCourse ORDER by char\_length(strCourCode) DESC, strCourCode DESC LIMIT 1;");

if (string.IsNullOrEmpty(strCode2))

{

txtCode2.Enabled = true;

}

else

{

strCode2 = Program.getNextCode(strCode2);

txtCode2.Text = strCode2;

txtCode2.Enabled = false;

}

txtCode2.BackColor = txtCode2.Enabled == true ? Color.White : Color.FromName("Control");

}

private void clearContents2()

{

txtAcronym2.Clear();

txtName2.Clear();

}

private void grid2\_CellMouseDoubleClick\_1(object sender, DataGridViewCellMouseEventArgs e)

{

if (grid2.CurrentRow != null)

{

txtCode2.Text = grid2.CurrentRow.Cells[0].Value.ToString();

txtName2.Text = grid2.CurrentRow.Cells[1].Value.ToString();

txtAcronym2.Text = grid2.CurrentRow.Cells[2].Value.ToString();

}

}

private int everythingIsOkay2()

{

if (grid.CurrentRow != null)

{

strCode1 = grid.CurrentRow.Cells[0].Value.ToString();

strCode2 = txtCode2.Text;

strAcronym2 = txtAcronym2.Text;

strName2 = txtName2.Text;

if (string.IsNullOrEmpty(strCode2) || string.IsNullOrEmpty(strAcronym2) || string.IsNullOrEmpty(strName2))

return 1;

}

else return 2;

return 0;

}

private void btnEdit2\_Click(object sender, EventArgs e)

{

if (grid2.CurrentRow != null)

{

txtCode2.Text = grid2.CurrentRow.Cells[0].Value.ToString();

txtName2.Text = grid2.CurrentRow.Cells[1].Value.ToString();

txtAcronym2.Text = grid2.CurrentRow.Cells[2].Value.ToString();

newRecord2 = false;

}

}

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

{

}

private void btnDelete2\_Click(object sender, EventArgs e)

{

if (grid2.CurrentRow != null)

{

if (MetroFramework.MetroMessageBox.Show(this.Parent.Parent, "Are you sure?", "DELETE CONFIRMATION", MessageBoxButtons.YesNo, MessageBoxIcon.Error) == DialogResult.Yes)

{

if (Program.ExecuteQuery("UPDATE tblCourse SET boolCourDeleted = TRUE, dtmCourDeleted=NOW(), strCourDeleted=CURRENT\_USER WHERE strCourCode = '" + grid2.CurrentRow.Cells[0].Value.ToString() + "';"))

{

query2 = "SELECT strCourCode AS 'Code', strCourName AS 'Course', IFNULL(strCourShort,'') AS 'Acronym' FROM tblCourse WHERE boolCourDeleted = FALSE AND strCourCollCode LIKE '"+grid.CurrentRow.Cells[0].Value.ToString()+"'";

updateTable2();

}

}

}

}

private void btnClear2\_Click(object sender, EventArgs e)

{

clearContents2();

}

private void btnSave2\_Click(object sender, EventArgs e)

{

switch (everythingIsOkay2())

{

case 0:

if (newRecord2)

query = "INSERT INTO `dbmedicalclinic`.`tblcourse` (`strCourCode`, `strCourName`, `strCourCollCode`, `strCourShort`, `strCourAdded`) VALUES ('" + strCode2 + "', '" + strName2 + "', '" + strCode1 + "', '" + strAcronym2 + "', USER());";

else

query = "UPDATE `dbmedicalclinic`.`tblcourse` SET `strCourName`= '"+strName2+"', `strCourCollCode`= '"+strCode1+"', `strCourShort`= '"+strAcronym2+"' WHERE `strCourCode`= '"+strCode2+"';";

if (!Program.ExecuteQuery(query))

{

if (newRecord2)

error("Failed to save course");

else

error("Failed to update course");

}

else

{

clearContents2();

status.Text = "Saved";

status.BackColor = Color.LimeGreen;

updateTable2();

getCode2();

}

break;

case 1:

error("Please fill out empty fields");

break;

case 2:

error("No college selected");

break;

}

}

private void btnAdd2\_Click(object sender, EventArgs e)

{

newRecord2 = true;

clearContents2();

getCode2();

txtName2.Focus();

}

private void grid2\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

}

#endregion

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

{

if (rb1.Checked)

{

query1 = "SELECT strCollCode AS 'Code', strCollName AS 'College', IFNULL(strCollShort,'') AS 'Acronym' FROM tblCollege WHERE boolCollDeleted=FALSE AND strCollName LIKE '%" + txtFilter.Text + "%';";

updateTable1();

}else

{

query2 = "SELECT strCourCode AS 'Code', strCourName AS 'Course', IFNULL(strCourShort,'') AS 'Acronym' FROM tblCourse WHERE strCourName LIKE '%" + txtFilter.Text + "%' AND boolCourDeleted=FALSE;";

updateTable2();

}

}

private void grid2\_CurrentCellChanged(object sender, EventArgs e)

{

}

private void error (string errmsg)

{

status.Text = errmsg;

status.BackColor = Color.Tomato;

}

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

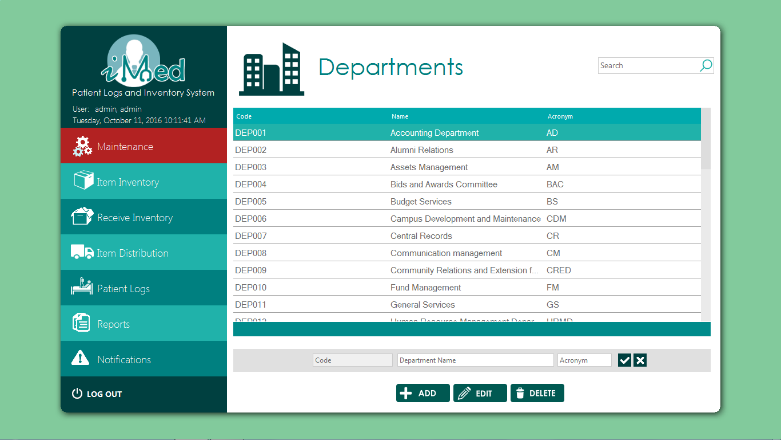
{

}

}

}

**Departments Form**

****

using MySql.Data.MySqlClient;

using System;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

namespace PUPiMed.UserControls

{

public partial class UCDepartment : UserControl

{

string query, strCode, strName, strAcronym;

bool newRecord;

public UCDepartment()

{

InitializeComponent();

query = "SELECT strDeptCode AS 'Code', strDeptName AS 'Name', strDeptShort AS 'Acronym' FROM tblDepartment WHERE boolDeptDeleted=FALSE AND strDeptCode LIKE '"+txtFilter.Text+"%'";

updateTable();

}

public void updateTable()

{

using(LoginNew.conn)

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (LoginNew.conn)

{

using (MySqlCommand cmd = new MySqlCommand(query, LoginNew.conn))

{

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

using (DataTable dt = new DataTable())

{

if (dt != null)

{

sda.Fill(dt);

grid.DataSource = dt;

}

}

}

}

}

}

}

private void txtFilter\_TextChanged(object sender, System.EventArgs e)

{

query = "SELECT strDeptCode AS 'Code', strDeptName AS 'Name', strDeptShort AS 'Acronym' FROM tblDepartment WHERE boolDeptDeleted=FALSE AND strDeptName LIKE '" + txtFilter.Text + "%'";

updateTable();

}

private void grid\_CurrentCellChanged(object sender, System.EventArgs e)

{

}

private void btnAdd\_Click(object sender, System.EventArgs e)

{

newRecord = true;

clearContents();

getCode();

txtName.Focus();

}

private void getCode()

{

strCode = Program.getPrevCode("SELECT strDeptCode from tblDepartment ORDER by char\_length(strDeptCode) DESC, strDeptCode DESC LIMIT 1;");

if (string.IsNullOrEmpty(strCode))

{

txtCode.Enabled = true;

}

else

{

strCode = Program.getNextCode(strCode);

txtCode.Text = strCode;

txtCode.Enabled = false;

}

txtCode.BackColor = txtCode.Enabled == true ? Color.White : Color.FromName("Control");

}

private bool everythingIsOkay()

{

getCode();

strCode = txtCode.Text;

strAcronym = txtAcronym.Text;

strName = txtName.Text;

if (string.IsNullOrEmpty(strCode) || string.IsNullOrEmpty(strAcronym) || string.IsNullOrEmpty(strName))

return false;

return true;

}

private void btnEdit1\_Click(object sender, EventArgs e)

{

if (grid.CurrentRow != null)

{

txtCode.Text = grid.CurrentRow.Cells[0].Value.ToString();

txtName.Text = grid.CurrentRow.Cells[1].Value.ToString();

txtAcronym.Text = grid.CurrentRow.Cells[2].Value.ToString();

newRecord = false;

}

}

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

{

clearContents();

}

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

{

if (everythingIsOkay())

{

if (newRecord)

query = "INSERT INTO `dbmedicalclinic`.`tbldepartment` (`strDeptCode`, `strDeptName`, `strDeptShort`, `strDeptAdded`) VALUES('" + strCode + "', '" + strName + "', '" + strAcronym + "', USER());";

else

query = "UPDATE `dbmedicalclinic`.`tbldepartment` SET `strDeptName`='"+strName+"', `strDeptShort`='"+strAcronym+"' WHERE `strDeptCode`='"+grid.CurrentRow.Cells[0].Value.ToString()+"';";

if (!Program.ExecuteQuery(query))

{

if (newRecord)

error("Failed to save department");

else

error("Failed to update department");

}

else

{

clearContents();

status.Text = "Saved";

status.BackColor = Color.LimeGreen;

query = "SELECT strDeptCode AS 'Code', strDeptName AS 'Name', strDeptShort AS 'Acronym' FROM tblDepartment WHERE boolDeptDeleted=FALSE AND strDeptCode LIKE '" + txtFilter.Text + "%'";

updateTable();

getCode();

}

}

else

{

error("Please fill out empty fields");

}

}

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

{

}

private void clearContents()

{

txtAcronym.Clear();

txtName.Clear();

}

private void error(string errmsg)

{

status.Text = errmsg;

status.BackColor = Color.Tomato;

}

private void btnDelete1\_Click(object sender, EventArgs e)

{

if (grid.CurrentRow != null)

{

if (MetroFramework.MetroMessageBox.Show(this.Parent.Parent, "Are you sure?", "DELETE CONFIRMATION", MessageBoxButtons.YesNo, MessageBoxIcon.Error) == DialogResult.Yes)

{

if (Program.ExecuteQuery("UPDATE tblDepartment SET boolDeptDeleted = TRUE, dtmDeptDeleted=NOW(), strDeptDeleted=CURRENT\_USER WHERE strDeptCode = '" + grid.CurrentRow.Cells[0].Value.ToString() + "';"))

{

query = "SELECT strDeptCode AS 'Code', strDeptName AS 'Name', strDeptShort AS 'Acronym' FROM tblDepartment WHERE boolDeptDeleted=FALSE AND strDeptCode LIKE '" + txtFilter.Text + "%'";

updateTable();

}

}

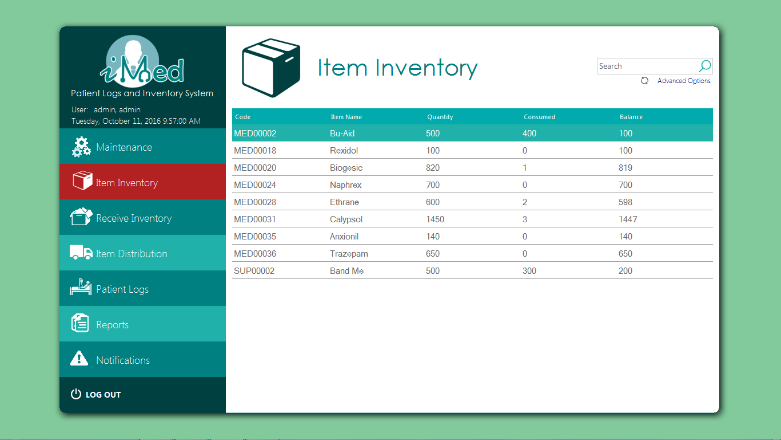
}

}

}

}

**Item Inventory Form**

****

using MySql.Data.MySqlClient;

using System;

using System.Data;

using System.Drawing;

using System.Windows.Forms;

namespace PUPiMed

{

public partial class UCItemInventory : UserControl

{

public string query =

"SELECT \* FROM vwstock;";

public UCItemInventory()

{

InitializeComponent();

updateTable(query, grid);

}

public void updateTable(string strQuery, DataGridView gridview)

{

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(strQuery, LoginNew.conn))

{

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

DataTable dt = new DataTable();

try

{

sda.Fill(dt);

gridview.DataSource = dt;

}

catch(Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

finally

{

dt.Dispose();

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

}

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

{

query = "SELECT \* FROM vwstock WHERE `Item Name` LIKE '" + txtFilter.Text + "%';";

updateTable(query, grid);

}

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

{

FormItemSearch form = new FormItemSearch(this);

query = "SELECT \* FROM vwstock WHERE `Item Name` LIKE '" + txtFilter.Text + "%' ";

form.ShowDialog();

}

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

{

foreach (DataGridViewRow row in grid.Rows)

{

//row.Cells[2].Style.BackColor = Color.Red;

}

}

string code;

private void grid\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

var row = grid.CurrentRow;

if (row != null)

{

if(grid.Height != 135 || row.Cells[1].Value.ToString()!=code)

{

grid.Height = 135;

gridPanel.Visible = true;

loadSubgrid(row.Cells[1].Value.ToString());

}

else

{

grid.Height = 520;

gridPanel.Visible = false;

}

}

}

private void loadSubgrid(string code)

{

this.code = code;

string queryIn = "SELECT \* from vwreceiveditem WHERE `Item Name` LIKE '"+code+"';";

string queryOut = "SELECT \* from vwdistribitem WHERE `Item Name` LIKE '" + code + "';";

string queryOut2 = "SELECT \* from vwlogsitem WHERE `Item Name` LIKE '" + code + "';";

updateTable(queryIn, gridIn);

updateTable(queryOut, gridOut);

updateTable(queryOut2, gridOut2);

}

private void grid\_Scroll(object sender, ScrollEventArgs e)

{

}

private void grid\_DataBindingComplete(object sender, DataGridViewBindingCompleteEventArgs e)

{

}

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

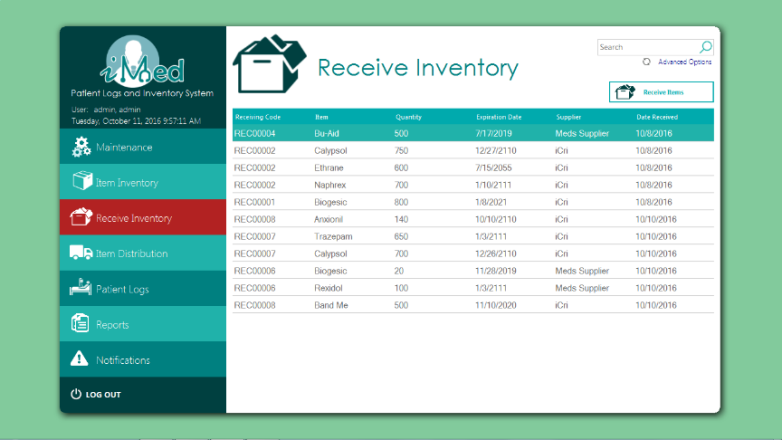
{

}

}

}

**Receive Inventory Form**

****

using System;

using System.Data;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace PUPiMed

{

public partial class UCReceive : UserControl

{

public string strQuery;

public UCReceive()

{

InitializeComponent();

strQuery = "SELECT \* FROM vwreceived WHERE `Item Name` LIKE '" + txtFilter.Text + "%' ORDER BY `Date Received`;";

updateTable();

}

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

{

new FormReceiveInventory(this).ShowDialog();

}

public void updateTable()

{

try

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(strQuery, LoginNew.conn))

{

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

DataTable dt = new DataTable();

try

{

sda.Fill(dt);

grid.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

finally

{

dt.Dispose();

}

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

}

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

{

strQuery = "SELECT \* FROM vwreceived WHERE `Item Name` LIKE '" + txtFilter.Text + "%' ORDER BY `Date Received`;";

updateTable();

}

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

{

new Forms.FormReceiveItemSearch(this).ShowDialog(); ;

}

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

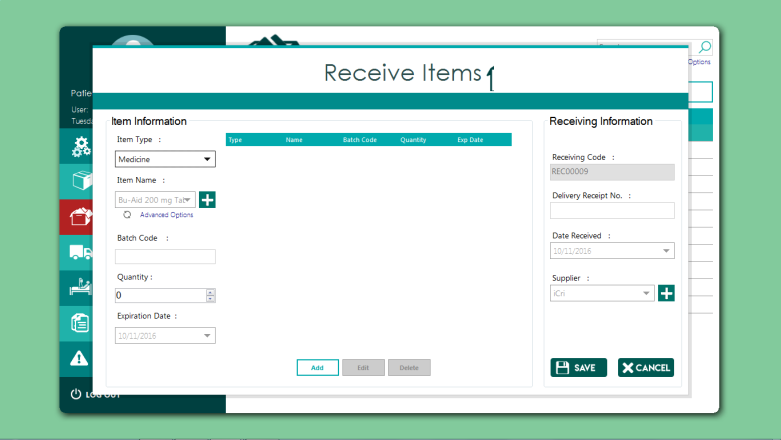
{

}

}

}

**Receivve Item Form**

******

using MetroFramework;

using MetroFramework.Controls;

using MetroFramework.Forms;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace PUPiMed

{

public partial class FormReceiveInventory : MetroForm

{

UCReceive parent;

List<String> listItemCode, listSuppCode;

string strType, strCode, strName, strQty, strSupplier, strExp, strRDate, strRCode, strDelivery;

public FormReceiveInventory(UCReceive parent)

{

this.parent = parent;

InitializeComponent();

dtReceived.Value = DateTime.Today;

cbType.SelectedIndex = 0;

loadItemComboBox(cbType.SelectedIndex+1);

loadSupplierComboBox();

icon.Parent = label1;

cbType.Focus();

}

private void error(string text)

{

status.Text = text;

status.BackColor = Color.Tomato;

}

private void clearStatus()

{

status.BackColor = Color.DarkCyan;

status.Text = "";

}

private void setMinMax()

{

txtQty.Minimum = Int32.Parse(Program.getPrevCode("SELECT CASE WHEN fnGetBalance(strItemCode) = intItemMin THEN 0 WHEN fnGetBalance(strItemCode) < intItemMin THEN intItemMin - fnGetBalance(strItemCode) WHEN fnGetBalance(strItemCode) > intItemMin THEN 0 END AS `Min` FROM tblItem WHERE boolItemDeleted = false AND strItemCode LIKE '"+listItemCode[cbName.SelectedIndex].ToString()+"';"));

txtQty.Maximum = Int32.Parse(Program.getPrevCode("SELECT CASE WHEN fnGetBalance(strItemCode) = intItemMax THEN 0 WHEN fnGetBalance(strItemCode) < intItemMax THEN intItemMax - fnGetBalance(strItemCode) WHEN fnGetBalance(strItemCode) > intItemMax THEN 0 END AS `Max` FROM tblItem WHERE boolItemDeleted = false AND strItemCode LIKE '" + listItemCode[cbName.SelectedIndex].ToString() + "';"));

}

public void loadItemComboBox(int type)

{ //add dosage and formulation for Meds

if (!Program.fillComboBox(cbName, "SELECT a.strItemCode, CONCAT(a.strItemName, ' ', IFNULL(b.strAddiDosage,''), ' ', IFNULL(b.strAddiForm,'')) AS Medicine FROM tblItem a LEFT JOIN tblAdditional b ON a.strItemCode = b.strAddiItemCode WHERE intItemType=" + type + " AND boolItemDeleted = false; ", out listItemCode))

{

pbAddItem.Focus();

cbName.PromptText = "Please add a new Item";

}

cbName.SelectedIndex = 0;

setMinMax();

}

public void loadSupplierComboBox()

{

if (!Program.fillComboBox(cbSupplier, "SELECT strSuppCode, strSuppName FROM tblSupplier;", out listSuppCode))

{

pbAddSupplier.Focus();

cbSupplier.PromptText = "Please add a new Supplier";

}

cbSupplier.SelectedIndex = 0;

}

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

{

}

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

{

this.parent.updateTable();

this.Dispose();

}

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

{

if (cbType != null)

{

if (cbType.SelectedItem.Equals("Medicine"))

new FormAddItem(1).ShowDialog();

else if (cbType.SelectedItem.Equals("Supply"))

new FormAddItem(2).ShowDialog();

else if (cbType.SelectedItem.Equals("Equipment"))

new FormAddItem(3).ShowDialog();

else

cbType.Focus();

loadItemComboBox(cbType.SelectedIndex + 1);

}

}

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

{

getCode();

}

private void getCode()

{

string prevCode = PUPiMed.Program.getPrevCode("SELECT strRecHCode from tblReceiveHeader ORDER by char\_length(strRecHCode) DESC, strRecHCode DESC LIMIT 1;");

if (string.IsNullOrEmpty(prevCode))

{

this.txtRCode.Enabled = true;

this.txtRCode.Enabled = true;

this.txtRCode.Focus();

this.txtRCode.BackColor = Color.White;

}

else

{

strRCode = PUPiMed.Program.getNextCode(prevCode);

this.txtRCode.Text = strRCode;

this.txtRCode.Enabled = false;

this.txtRCode.BackColor = Color.FromArgb(224, 224, 224);

this.cbType.Focus();

}

}

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

{

if (headerIsOkay())

{

if (listDetail.Rows.Count>0)

{

if (saveHeader())

{

if (saveDetail())

{

clearFields();

error("Saved");

status.BackColor = Color.LimeGreen;

}

}

}

else

{

error("Please add an item first");

}

}

else

{

error("Please fill out the required fields before saving");

}

}

private bool saveHeader()

{

string strQuery = "INSERT INTO tblreceiveheader (`strRecHCode`, `strRecHDeliveryReceipt`, `strRecHSuppCode`, `datRecHDateReceived`, `strRecHAdded`) VALUES('" + strRCode+"','"+strDelivery+"','" + listSuppCode[cbSupplier.SelectedIndex]+"','"+strRDate+"', USER());";

if (!PUPiMed.Program.ExecuteQuery(strQuery))

{

return false;

}

return true;

}

private bool saveDetail()

{

string strQuery = "";

foreach (DataGridViewRow row in listDetail.Rows)

{

loadItemComboBox(1 + cbType.FindStringExact(row.Cells[0].Value.ToString()));

strQuery =

"INSERT INTO tblreceivedetail (`strRecDRecHCode`, `strRecDItemCode`, `strRecDBatchCode`, `intRecDQty`, `datRecDExpDate`) VALUES ('"

+ strRCode + "','" //Receiving Code

+ listItemCode[cbName.FindStringExact(row.Cells[1].Value.ToString())].ToString() + "','" //itemCode

+ row.Cells[2].Value.ToString() + "','" //BatchCode

+ row.Cells[3].Value.ToString() + "','" //QTY

+ row.Cells[4].Value.ToString() + "');"; //DATE EXP

if (!PUPiMed.Program.ExecuteQuery(strQuery))

{

return false;

}

}

return true;

}

//edit row

private void metroButton2\_Click(object sender, EventArgs e)

{

foreach (DataGridViewRow row in listDetail.SelectedRows)

{

cbType.SelectedItem = row.Cells[0].Value.ToString();

cbName.SelectedItem = row.Cells[1].Value.ToString();

txtCode.Text = row.Cells[2].Value.ToString();

txtQty.Text = row.Cells[3].Value.ToString();

dtExp.Value = DateTime.Parse(row.Cells[4].Value.ToString());

listDetail.Rows.Remove(row);

}

}

private void materialLabel5\_Click(object sender, EventArgs e)

{

}

private void label5\_Click(object sender, EventArgs e)

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void materialDivider1\_Click(object sender, EventArgs e)

{

}

private void label4\_Click(object sender, EventArgs e)

{

}

private void label6\_Click(object sender, EventArgs e)

{

}

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

{

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);

}

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

{

clearStatus();

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);

}

private void dtReceived\_ValueChanged(object sender, EventArgs e)

{

dtReceived.MaxDate = DateTime.Today;

}

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

{

}

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

{

clearStatus();

setMinMax();

}

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

{

clearStatus();

}

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

{

clearStatus();

}

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

{

clearStatus();

}

private void dtReceived\_DropDown(object sender, EventArgs e)

{

clearStatus();

}

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

{

clearStatus();

}

private void txtQty\_Validated(object sender, EventArgs e)

{

}

private void txtCode\_EnabledChanged(object sender, EventArgs e)

{

MetroTextBox txt = sender as MetroTextBox;

if (!txt.Enabled)

txt.BackColor = Color.Gainsboro;

else

txt.BackColor = Color.White;

}

private void listReceived\_ItemActivate(object sender, EventArgs e)

{

if(listDetail.SelectedRows.Count > 0) {

btnEdit.Enabled = true;

btnDelete.Enabled = true;

}else

{

btnEdit.Enabled = false;

btnDelete.Enabled = false;

}

}

private void dtExp\_ValueChanged(object sender, EventArgs e)

{

clearStatus();

}

private void clearFields()

{

cbType.SelectedIndex = 0;

loadItemComboBox(1);

loadSupplierComboBox();

txtQty.Text = "0";

getCode();

dtExp.Value = DateTime.Today;

dtReceived.Value = DateTime.Today;

listDetail.Rows.Clear();

txtCode.Clear();

txtDeliveryNo.Clear();

cbType.Focus();

}

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

{

new FormAddSupplier(this).ShowDialog();

}

private bool headerIsOkay()

{

strRCode = txtRCode.Text;

strDelivery = txtDeliveryNo.Text;

strRDate = dtReceived.Value.ToString("yyyy-MM-dd");

strSupplier = cbSupplier.SelectedItem.ToString();

if (string.IsNullOrWhiteSpace(strRCode) || string.IsNullOrWhiteSpace(strDelivery) || string.IsNullOrWhiteSpace(strRDate) || string.IsNullOrWhiteSpace(strSupplier))

{

return false;

}

return true;

}

private bool detailIsOkay()

{

strType = cbType.SelectedItem.ToString();

strCode = txtCode.Text;

strName = cbName.SelectedItem.ToString();

strQty = txtQty.Text;

strExp = dtExp.Value.ToString("yyyy-MM-dd");

if (string.IsNullOrWhiteSpace(strType) || string.IsNullOrWhiteSpace(strName)|| string.IsNullOrWhiteSpace(strQty) || string.IsNullOrWhiteSpace(strExp))

{

return false;

}else

{

if (cbType.SelectedIndex == 0 && string.IsNullOrWhiteSpace(strCode))

return false;

}

return true;

}

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

{

}

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

{

string[] row = new string[5];

int qty;

if (!detailIsOkay())

{

error("Please fill out the required fields before adding");

}

else

{

if ((Int32.TryParse(strQty, out qty)))

{

//qty cant be zero or less

if (qty <= 0)

{

error("Quantity can't be less than one");

}

else

{

if (!PUPiMed.Program.duplicateEntry(listDetail, strName, 1))

{

row[0] = strType;

row[1] = strName;

row[2] = strCode;

row[3] = strQty;

row[4] = strExp;

listDetail.Rows.Add(row);

//cbType.SelectedIndex = 0;

cbName.SelectedIndex = 0;

txtCode.Clear();

txtQty.Text = "0";

dtExp.Value = DateTime.Today;

cbType.Focus();

}

else

{

error("Duplicate entry for " + strName);

}

}

}

else

{

error("Please enter a numeric value for 'Quantity'");

}

if (listDetail.SelectedRows.Count > 0)

{

btnEdit.Enabled = true;

btnDelete.Enabled = true;

}

else

{

btnEdit.Enabled = false;

btnDelete.Enabled = false;

}

}

}

//Delete row

private void metroButton1\_Click(object sender, EventArgs e)

{

foreach (DataGridViewRow row in listDetail.SelectedRows)

{

listDetail.Rows.Remove(row);

}

if (listDetail.SelectedRows.Count > 0)

{

btnEdit.Enabled = true;

btnDelete.Enabled = true;

}

else

{

btnEdit.Enabled = false;

btnDelete.Enabled = false;

}

}

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

{

clearStatus();

loadItemComboBox(cbType.SelectedIndex + 1);

//string str = "";

//for(int i = 0; i < alistCode.Count; i++)

//{

// str += alistCode[i] + " - " + cbName.Items[i] + "\n";

//}

//MessageBox.Show(str);

}

private bool dragging = false;

private Point dragCursorPoint;

private Point dragFormPoint;

private void pictureBox1\_MouseDown(object sender, MouseEventArgs e)

{

dragging = true;

dragCursorPoint = Cursor.Position;

dragFormPoint = this.Location;

}

private void pictureBox1\_MouseMove(object sender, MouseEventArgs e)

{

if (dragging)

{

Point dif = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));

this.Location = Point.Add(dragFormPoint, new Size(dif));

}

}

private void pictureBox1\_MouseUp(object sender, MouseEventArgs e)

{

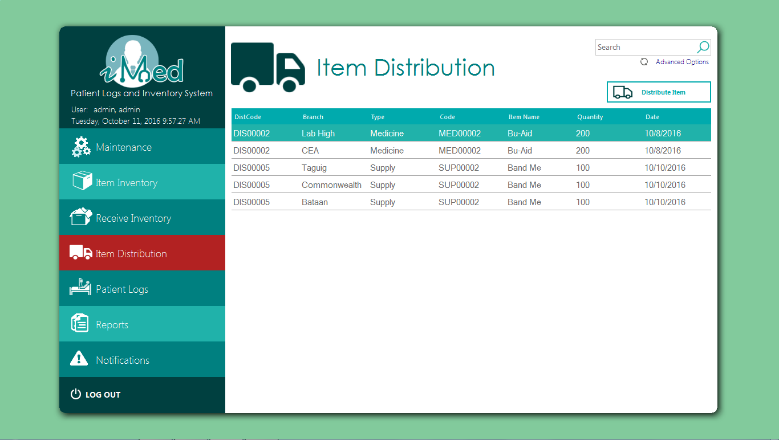
dragging = false;

}

}

}

**Item Distribution From**

****

using MySql.Data.MySqlClient;

using System;

using System.Data;

using System.Windows.Forms;

namespace PUPiMed

{

public partial class UCItemDistribution : UserControl

{

public string query = "SELECT \* FROM vwDistrib;";

public UCItemDistribution()

{

InitializeComponent();

updateTable();

}

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

{

}

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

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void pictureBox12\_Click(object sender, EventArgs e)

{

}

public void updateTable()

{

try

{

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(query, LoginNew.conn))

{

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

DataTable dt = new DataTable();

try

{

sda.Fill(dt);

grid.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

finally

{

dt.Dispose();

}

}

}

}catch(Exception exx)

{

MessageBox.Show(exx.Message.ToString());

}

finally

{

LoginNew.conn.Close();

}

}

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

{

new FormAddDistribItem(this).ShowDialog();

}

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

{

FormItemSearch form = new FormItemSearch(this);

query = "SELECT \* FROM vwDistrib WHERE `Item Name` LIKE '" + txtFilter.Text + "%' ";

form.ShowDialog();

}

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

{

query = "SELECT \* FROM vwDistrib WHERE `Item Name` LIKE '" + txtFilter.Text + "%';";

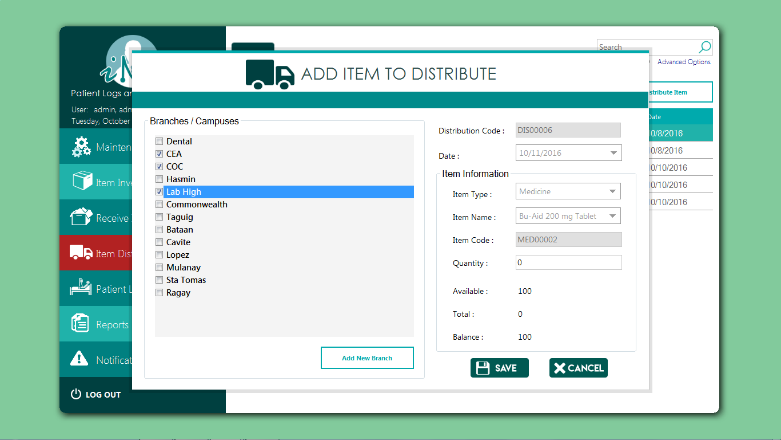
updateTable();

}

}

}

**Add Item to Distribute Form**

****

using MetroFramework;

using MetroFramework.Forms;

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace PUPiMed

{

public partial class FormAddDistribItem : MetroForm

{

UCItemDistribution parent;

List<String> listCode;

List<string> campus = new List<string>();

List<string> campuscode = new List<string>();

string strType, strCode, strName, strQty, strDCode, strDate;

int intBalance, intTotal, intAvailable, intQty;

public FormAddDistribItem(UCItemDistribution parent)

{

InitializeComponent();

getCode();

this.parent = parent;

cbType.SelectedIndex = 0;

cbType.Focus();

loadBranches();

dtDate.Value = DateTime.Today;

}

private void getBalance()

{

intAvailable = int.Parse(PUPiMed.Program.getPrevCode("SELECT IFNULL(fnGetBalance('" + strCode + "'),0);"));

txtAvailable.Text = intAvailable.ToString();

getIntValues();

}

private void getIntValues()

{

strQty = txtQty.Text;

if (!Int32.TryParse(strQty, out intQty))

{

txtQty.Text = "0";

}

getSelectedCampus();

int count = campus.Count;

intTotal = intQty \* count;

intBalance = intAvailable - intTotal;

txtBalance.Text = intBalance.ToString();

txtTotal.Text = intTotal.ToString();

}

private void getSelectedCampus()

{

campus.Clear();

foreach (int index in this.cbBranch.CheckedIndices)

{

campus.Add(campuscode[index]);

}

}

private void clearItems()

{

txtQty.Text = "0";

txtBalance.Text = "0";

txtAvailable.Text = "0";

txtTotal.Text = "0";

foreach (int index in this.cbBranch.CheckedIndices)

{

cbBranch.SetItemChecked(index, false);

}

foreach (var cb in this.Controls.OfType<ComboBox>())

{

cb.SelectedIndex = 0;

}

dtDate.Value = DateTime.Today;

getBalance();

}

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

{

}

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

{

getBalance();

}

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

{

if (everythingIsOkay())

{

if (intBalance >= 0)

{

if (cbBranch.CheckedItems.Count > 0)

{

if (int.Parse(strQty) > 0)

{

if (saveHeader())

{

if (saveCampusDetail())

{

clearItems();

getCode();

status.Text = "Saved.";

status.BackColor = Color.LimeGreen;

}

}

}

else

error("Quantity can't be zero");

}

else

error("Please select branches before saving");

}

else

error("Insufficient stock");

}

else

error("Please fill out the required fields before saving");

}

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

{

getIntValues();

}

private void error (string text)

{

status.Text = text;

status.BackColor = Color.Tomato;

}

private void clearStatus()

{

status.BackColor = Color.DarkCyan;

status.Text = "";

}

private void materialCheckBox5\_CheckStateChanged(object sender, EventArgs e)

{

getIntValues();

}

private void metroLabel8\_Click(object sender, EventArgs e)

{

}

private void metroLabel1\_Click(object sender, EventArgs e)

{

}

private void metroLabel2\_Click(object sender, EventArgs e)

{

}

private void metroLabel4\_Click(object sender, EventArgs e)

{

}

private void metroTextBox1\_Click(object sender, EventArgs e)

{

}

private void dtDate\_ValueChanged(object sender, EventArgs e)

{

dtDate.MaxDate = DateTime.Today;

}

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

{

this.parent.updateTable();

this.Dispose();

}

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

{

}

private void cbBranch\_ItemCheck(object sender, ItemCheckEventArgs e)

{

getIntValues();

}

private void cbBranch\_MouseClick(object sender, MouseEventArgs e)

{

getIntValues();

}

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

{

if(!String.IsNullOrWhiteSpace(txtQty.Text)|| !String.IsNullOrEmpty(txtQty.Text))

getIntValues();

}

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

{

if (!String.IsNullOrWhiteSpace(txtQty.Text) || !String.IsNullOrEmpty(txtQty.Text))

getIntValues();

}

private void cbBranch\_MouseEnter(object sender, EventArgs e)

{

if (!String.IsNullOrWhiteSpace(txtQty.Text) || !String.IsNullOrEmpty(txtQty.Text))

getIntValues();

}

private void cbBranch\_MouseClick\_1(object sender, MouseEventArgs e)

{

if (!String.IsNullOrWhiteSpace(txtQty.Text) || !String.IsNullOrEmpty(txtQty.Text))

getIntValues();

}

private void cbBranch\_ItemCheck(object sender, EventArgs e)

{

if (!String.IsNullOrWhiteSpace(txtQty.Text) || !String.IsNullOrEmpty(txtQty.Text))

getIntValues();

}

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

{

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);

}

private void groupBox2\_Enter(object sender, EventArgs e)

{

}

private void metroLabel3\_Click(object sender, EventArgs e)

{

}

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

{

}

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

{

new Forms.FormAddBranches(this).ShowDialog();

}

private void getCode()

{

string prevCode = PUPiMed.Program.getPrevCode("SELECT strDisHCode from tblDistributionHeader ORDER by char\_length(strDisHCode) DESC, strDisHCode DESC LIMIT 1;");

if (string.IsNullOrEmpty(prevCode))

{

this.txtDCode.Enabled = true;

this.txtDCode.Focus();

this.txtDCode.BackColor = Color.White;

}

else

{

strDCode = PUPiMed.Program.getNextCode(prevCode);

this.txtDCode.Text = strDCode;

this.txtDCode.Enabled = false;

this.txtDCode.BackColor = Color.FromArgb(224, 224, 224);

}

}

private bool everythingIsOkay()

{

strDCode = txtDCode.Text;

strType = cbType.SelectedItem.ToString();

strCode = txtCode.Text;

strName = cbName.SelectedItem.ToString();

strQty = txtQty.Text;

if (!Int32.TryParse(strQty, out intQty))

{

error("Please enter a numeric value for Quantity");

return false;

}

strDate = dtDate.Value.ToString("yyyy-MM-dd");

if (string.IsNullOrWhiteSpace(strDCode) || string.IsNullOrWhiteSpace(strDate)

|| string.IsNullOrWhiteSpace(strType) || string.IsNullOrWhiteSpace(strName) || string.IsNullOrWhiteSpace(strCode))

{

return false;

}

return true;

}

private bool saveHeader()

{

string strQuery =

"INSERT INTO tblDistributionHeader (`strDisHCode`, `datDisHDate`, `strDisHItemCode`, `intDisHQty`, `strDisHAdded`) VALUES ('"

+ strDCode + "', '"

+ strDate + "', '"

+ strCode + "', "

+ strQty + ",USER());";

if (!PUPiMed.Program.ExecuteQuery(strQuery))

{

error("Failed to save to database");

MetroMessageBox.Show(this, "Failed to save to database.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return false;

}

return true;

}

private bool saveCampusDetail()

{

getSelectedCampus();

string strQuery = "";

foreach(string strBranch in campus)

{

strQuery=

"INSERT INTO tbldistributiondetail VALUES " +

"('" + strDCode + "', '" + strBranch + "');";

if (!PUPiMed.Program.ExecuteQuery(strQuery))

{

return false;

}

}

return true;

}

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

{

if (cbName.SelectedIndex <= listCode.Count && listCode != null)

{

txtCode.Text = listCode[cbName.SelectedIndex].ToString();

strCode = txtCode.Text;

getBalance();

getIntValues();

}

}

private void loadItemComboBox(int type)

{

if (!Program.fillComboBox(cbName, "SELECT a.strItemCode, CONCAT(a.strItemName, ' ', IFNULL(b.strAddiDosage,''), ' ', IFNULL(b.strAddiForm,'')) AS Medicine FROM tblItem a LEFT JOIN tblAdditional b ON a.strItemCode = b.strAddiItemCode WHERE fnGetBalance(strItemCode)>intItemMin AND intItemType=" + type + " AND boolItemDeleted = false; ", out listCode))

{

cbName.Enabled = false;

txtCode.Text = "";

error("No available item");

}

else

{

cbName.Enabled = true;

txtCode.Text = listCode[0].ToString();

}

cbName.SelectedIndex = 0;

clearStatus();

}

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

{

}

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

{

loadItemComboBox(cbType.SelectedIndex+1);

}

public void loadBranches()

{

cbBranch.Items.Clear();

try

{

if (LoginNew.conn != null && LoginNew.conn.State == ConnectionState.Closed)

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand("SELECT \* FROM tblbranch where boolbrandeleted=false;", LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

while (dataReader.Read())

{

campuscode.Add(dataReader.GetString(0));

cbBranch.Items.Add(dataReader.GetString(1));

}

dataReader.Close();

}

}

}

catch (Exception ex) {

MessageBox.Show(ex.Message);

}

finally

{

LoginNew.conn.Close();

}

}

private bool dragging = false;

private Point dragCursorPoint;

private Point dragFormPoint;

private void pictureBox1\_MouseDown(object sender, MouseEventArgs e)

{

dragging = true;

dragCursorPoint = Cursor.Position;

dragFormPoint = this.Location;

}

private void pictureBox1\_MouseMove(object sender, MouseEventArgs e)

{

if (dragging)

{

Point dif = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));

this.Location = Point.Add(dragFormPoint, new Size(dif));

}

}

private void pictureBox1\_MouseUp(object sender, MouseEventArgs e)

{

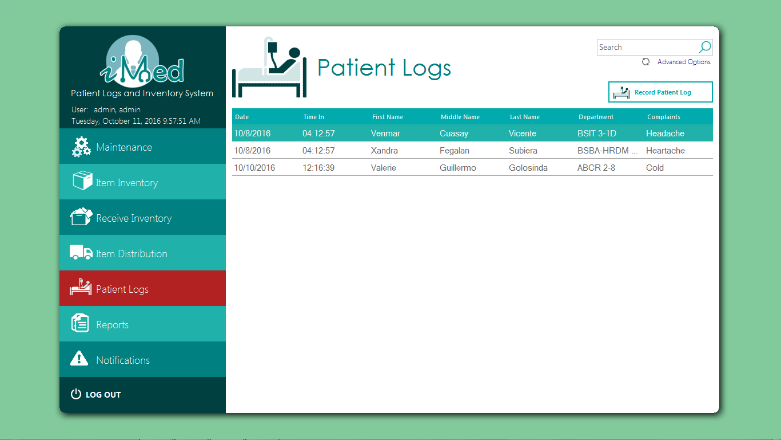
dragging = false;

}

}

}

**Patient Logs Form**

****

using System;

using System.Data;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace PUPiMed

{

public partial class UCPatientLogs\_DTR\_ : UserControl

{

public string strQuery;

public UCPatientLogs\_DTR\_()

{

InitializeComponent();

strQuery = "SELECT `Date`, `Time In`, `First Name`, `Middle Name`, `Last Name`, `Department`, `Complaints` FROM vwlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

updateTable();

}

public void updateTable()

{

try

{

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand(strQuery, LoginNew.conn))

{

using (MySqlDataAdapter sda = new MySqlDataAdapter(cmd))

{

DataTable dt = new DataTable();

try

{

sda.Fill(dt);

grid.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

finally

{

dt.Dispose();

}

}

}

}

catch (Exception exx)

{

MessageBox.Show(exx.Message.ToString());

}

finally

{

LoginNew.conn.Close();

}

}

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

{

}

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

{

new FormAddPatientLogs(this).ShowDialog();

}

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

{

strQuery = "SELECT `Date`, `Time In`, `First Name`, `Middle Name`, `Last Name`, `Department`, `Complaints` FROM vwlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%' ";

new FormPatientSearch(this).ShowDialog();

}

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

{

strQuery = "SELECT `Date`, `Time In`, `First Name`, `Middle Name`, `Last Name`, `Department`, `Complaints` FROM vwlogs WHERE `Last Name` LIKE '" + txtFilter.Text + "%' OR `Middle Name` LIKE '" + txtFilter.Text + "%' OR `First Name` LIKE '" + txtFilter.Text + "%';";

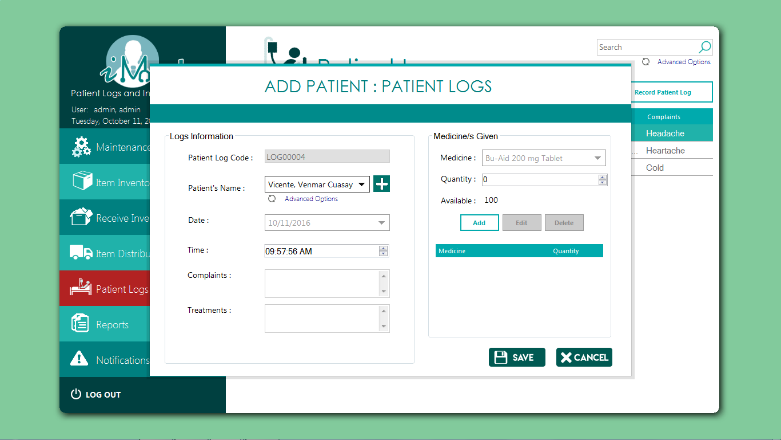
updateTable();

}

}

}

**Add Patient Log Form**

****

using MetroFramework.Forms;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace PUPiMed

{

public partial class FormAddPatientLogs : MetroForm

{

string strCode, strName, strTime, strDate, strComplaint, strTreatment, strQty, strMedGiven;

public string strQuery;

List<String> listMed, listName;

private UCPatientLogs\_DTR\_ uCPatientLogs\_DTR\_;

private int intAvailable;

public FormAddPatientLogs()

{

InitializeComponent();

dtTime.Value = DateTime.Now;

dtTime.MaxDate = DateTime.Now;

loadMedicine();

loadPatients("SELECT strPatiCode, Concat(strPatiLastName, ', ', strPatiFirstName, ' ', strPatiMiddleName) FROM tblPatient WHERE boolPatiDeleted = FALSE");

getCode();

}

public FormAddPatientLogs(UCPatientLogs\_DTR\_ uCPatientLogs\_DTR\_): this()

{

this.uCPatientLogs\_DTR\_ = uCPatientLogs\_DTR\_;

}

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

{

new FormAddPatient(this).ShowDialog();

}

private void dtTimeIn\_ValueChanged\_1(object sender, EventArgs e)

{

clearStatus();

dtTime.Format = DateTimePickerFormat.Custom;

dtTime.CustomFormat = "hh:mm:ss tt";

dtTime.MaxDate = DateTime.Now;

dtTime.ShowUpDown = true;

}

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

{

if (headerIsOkay())

{

if (saveHeader())

{

if (listMedsGiven.Rows.Count > 0)

{

if (!saveDetail())

{

error("Failed to save Log Detail");

}

else

{

getCode();

cbName.SelectedIndex = 0;

dtDate.Value = DateTime.Now.Date;

txtComplaints.Clear();

txtTreatments.Clear();

listMedsGiven.Rows.Clear();

cbMedsGiven.SelectedIndex = 0;

txtQty.Value = 0;

if (uCPatientLogs\_DTR\_ != null)

{

uCPatientLogs\_DTR\_.updateTable();

}

error("Saved");

status.BackColor = Color.LimeGreen;

getBalance();

}

}

}

}

else

{

error("Please fill out the required fields before saving");

}

}

private void toolStripMenuItem2\_Click(object sender, EventArgs e)

{

}

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

{

}

private void error(string text)

{

status.Text = text;

status.BackColor = Color.Tomato;

}

private void clearStatus()

{

status.BackColor = Color.DarkCyan;

status.Text = "";

}

private bool saveHeader()

{

string strQuery = "INSERT INTO tblLogsHeader (`strLogHCode`, `datLogHDatetime`, `strLogHPatiCode`, `strLogsComplaints`, `strLogHTreatment`, strLogHAdded) VALUES('" + strCode + "','" + strDate+strTime + "','" + listName[cbName.SelectedIndex].ToString() + "','" + strComplaint + "','" + strTreatment + "', USER());";

if (!Program.ExecuteQuery(strQuery))

{

return false;

}

return true;

}

private bool headerIsOkay()

{

strCode = txtCode.Text;

strName = cbName.SelectedItem.ToString();

strTime = dtTime.Value.ToString("hh-mm-ss");

strDate = dtDate.Value.ToString("yyyy-MM-dd-");

strComplaint = txtComplaints.Text.ToString();

strTreatment = txtTreatments.Text.ToString();

if (string.IsNullOrWhiteSpace(strCode) ||

string.IsNullOrWhiteSpace(strName) ||

string.IsNullOrWhiteSpace(strTime) ||

string.IsNullOrWhiteSpace(strDate) ||

string.IsNullOrWhiteSpace(strComplaint) ||

string.IsNullOrWhiteSpace(strTreatment)

){

return false;

}

return true;

}

private bool detailIsOkay()

{

strMedGiven = cbMedsGiven.SelectedItem.ToString();

strQty = txtQty.Text;

if (string.IsNullOrEmpty(strMedGiven) || string.IsNullOrEmpty(strQty))

{

return false;

}else

{

if (Int32.Parse(strQty) < 1)

return false;

}

return true;

}

private bool saveDetail()

{

string strQuery;

foreach (DataGridViewRow row in listMedsGiven.Rows)

{

strQuery =

"INSERT INTO tblLogsDetail VALUES ('"

+ strCode + "','"

+ listMed[cbMedsGiven.FindStringExact(row.Cells[0].Value.ToString())].ToString() + "','"

+ strQty

+ "');";

if (!Program.ExecuteQuery(strQuery))

{

return false;

}

}

return true;

}

private void metroButton1\_Click(object sender, EventArgs e) //ADD

{

if (detailIsOkay())

{

if (!Program.duplicateEntry(listMedsGiven, strMedGiven , 0))

{

string[] row = new string[2];

row[0] = strMedGiven;

row[1] = strQty;

listMedsGiven.Rows.Add(row);

cbMedsGiven.SelectedIndex = 0;

txtQty.Value = 0;

}

else

{

error("Duplicate entry for "+strMedGiven);

}

}

else

{

error("Quantity can't be zero");

}

}

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

{

foreach (DataGridViewRow row in listMedsGiven.SelectedRows)

{

listMedsGiven.Rows.Remove(row);

}

if (listMedsGiven.SelectedRows.Count > 0)

{

btnEdit.Enabled = true;

btnDelete.Enabled = true;

}

else

{

btnEdit.Enabled = false;

btnDelete.Enabled = false;

}

}

private void listMedsGiven\_SelectionChanged(object sender, EventArgs e)

{

if (listMedsGiven.SelectedRows.Count > 0)

{

btnEdit.Enabled = true;

btnDelete.Enabled = true;

}

else

{

btnEdit.Enabled = false;

btnDelete.Enabled = false;

}

}

private void listMedsGiven\_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)

{

if (listMedsGiven.SelectedRows.Count > 0)

{

btnEdit.Enabled = true;

btnDelete.Enabled = true;

}

else

{

btnEdit.Enabled = false;

btnDelete.Enabled = false;

}

}

private void listMedsGiven\_RowsRemoved(object sender, DataGridViewRowsRemovedEventArgs e)

{

if (listMedsGiven.SelectedRows.Count > 0)

{

btnEdit.Enabled = true;

btnDelete.Enabled = true;

}

else

{

btnEdit.Enabled = false;

btnDelete.Enabled = false;

}

}

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

{

clearStatus();

}

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

{

clearStatus();

getBalance();

txtQty.Maximum = Int32.Parse(txtAvailable.Text);

txtQty.Value = txtQty.Minimum;

}

private void dtDate\_ValueChanged(object sender, EventArgs e)

{

clearStatus();

}

private void metroLabel7\_Click(object sender, EventArgs e)

{

}

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

{

loadPatients(null);

}

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

{

foreach (DataGridViewRow row in listMedsGiven.SelectedRows)

{

listMedsGiven.Rows.Remove(row);

cbMedsGiven.SelectedItem = row.Cells[0].Value.ToString();

txtQty.Text = row.Cells[1].Value.ToString();

}

}

private void getBalance()

{

try

{

intAvailable = int.Parse(PUPiMed.Program.getPrevCode("SELECT fnGetBalance('" + listMed[cbMedsGiven.SelectedIndex].ToString() + "');"));

txtAvailable.Text = intAvailable.ToString();

}

catch { }

}

private void txtQty\_ValueChanged(object sender, ScrollEventArgs e)

{

}

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

{

}

public void loadMedicine()

{

if (!Program.fillComboBox(cbMedsGiven, "SELECT a.strItemCode, CONCAT(a.strItemName, ' ', IFNULL(b.strAddiDosage,''), ' ', IFNULL(b.strAddiForm,'')) AS Medicine FROM tblItem a LEFT JOIN tblAdditional b ON a.strItemCode = b.strAddiItemCode WHERE fnGetBalance(strItemCode)>0 AND intItemType=1 AND boolItemDeleted = false; ", out listMed))

{

cbMedsGiven.PromptText = "No Medicine Found!";

}

cbMedsGiven.SelectedIndex = 0;

}

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

{

this.Dispose();

}

public void loadPatients(string query)

{

if (string.IsNullOrEmpty(query))

{

loadPatients("SELECT strPatiCode AS `Code`, Concat(strPatiLastName, ', ', strPatiFirstName, ' ', strPatiMiddleName) FROM tblPatient WHERE boolPatiDeleted = FALSE; ");

}else

{

if (!Program.fillComboBox(cbName, query, out listName))

{

cbName.PromptText = "No Patient Found!";

}

cbName.SelectedIndex = 0;

}

}

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

{

FormPatientSearch form = new FormPatientSearch(this);

form.ShowDialog();

}

private void getCode()

{

string prevCode = Program.getPrevCode("SELECT strLogHCode from tblLogsHeader ORDER by char\_length(strLogHCode) DESC, strLogHCode DESC LIMIT 1;");

if (string.IsNullOrEmpty(prevCode))

{

this.txtCode.Enabled = true;

this.txtCode.BackColor = Color.White;

this.txtCode.Focus();

}

else

{

strCode = Program.getNextCode(prevCode);

this.txtCode.Text = strCode;

this.txtCode.Enabled = false;

this.txtCode.BackColor = Color.FromArgb(224,224,224);

this.cbName.Focus();

}

}

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

{

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);

}

private bool everythingIsOkay()

{

if (!headerIsOkay()||listMedsGiven.Rows.Count<1)

{

return false;

}

return true;

}

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

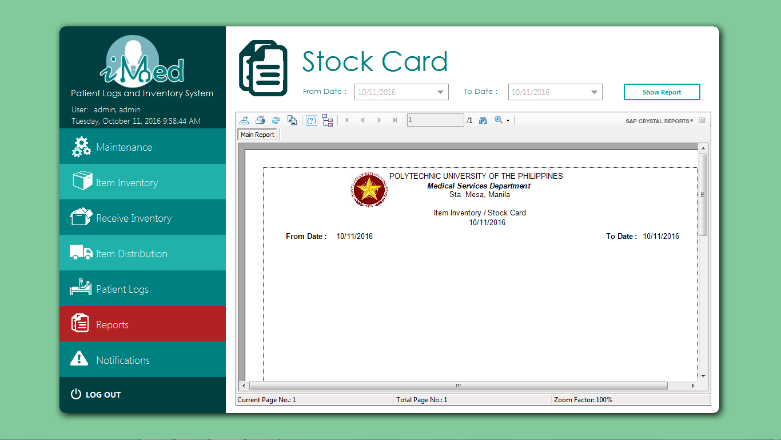
{

}

}

}

**Stock Card Form (Report)**

****

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using CrystalDecisions.Shared;

using System.Data.SqlClient;

namespace PUPiMed.UserControls

{

public partial class UCReportStockCard : UserControl

{

public UCReportStockCard()

{

InitializeComponent();

dtTo.Value = DateTime.Today;

dtTo.MaxDate = DateTime.Today;

dtFrom.Value = DateTime.Today;

dtFrom.MaxDate = DateTime.Today;

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptStocks.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = dtFrom.Value.ToString();

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["dateStart"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = dtTo.Value.ToString();

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["dateEnd"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer4.ReportSource = cryRpt;

crystalReportViewer4.Refresh();

}

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

{

}

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

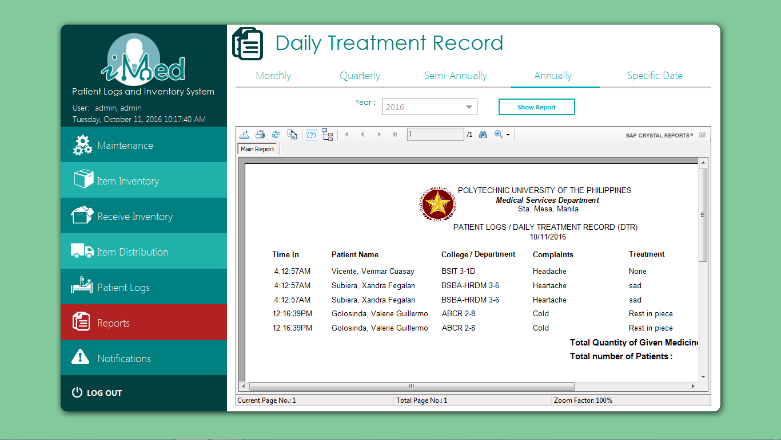
{

}

}

}

**Daily Treatment Record Form (Record)**

****

using System;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using CrystalDecisions.Shared;

using System.Data.SqlClient;

namespace PUPiMed

{

public partial class UCReports : UserControl

{

public UCReports()

{

InitializeComponent();

int year = Convert.ToInt32(DateTime.Today.Year.ToString());

for (int x = year; x != year - 20; x--)

{

cbMoYear.Items.Add(x);

cbYear.Items.Add(x);

cbSemiYear.Items.Add(x);

cbAnnualYear.Items.Add(x);

}

cbMoMonth.SelectedIndex = 0;

cbMoYear.SelectedIndex = 0;

//monthly();

cbYear.SelectedIndex = 0;

cbMonth.SelectedIndex = 0;

//quart();

cbSemiMonth.SelectedIndex = 0;

cbSemiYear.SelectedIndex = 0;

//semi();

cbAnnualYear.SelectedIndex = 0;

//annual();

dtTo.MaxDate = DateTime.Today;

}

private void monthly()

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDTRMonthly.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbMoMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbMoYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer5.ReportSource = cryRpt;

crystalReportViewer5.Refresh();

}

private void quart()

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDTRQuarterly.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer1.ReportSource = cryRpt;

crystalReportViewer1.Refresh();

}

private void semi()

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDTRSemi.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbSemiMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbSemiYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer2.ReportSource = cryRpt;

crystalReportViewer2.Refresh();

}

private void annual()

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDTRAnnual.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbAnnualYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer3.ReportSource = cryRpt;

crystalReportViewer3.Refresh();

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDTRPickADate.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = dtFrom.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["fromDate"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = dtTo.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["toDate"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer4.ReportSource = cryRpt;

crystalReportViewer4.Refresh();

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDTRMonthly.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbMoMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbMoYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer5.ReportSource = cryRpt;

crystalReportViewer5.Refresh();

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDTRQuarterly.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer1.ReportSource = cryRpt;

crystalReportViewer1.Refresh();

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDTRSemi.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbSemiMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbSemiYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer2.ReportSource = cryRpt;

crystalReportViewer2.Refresh();

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDTRAnnual.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbAnnualYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer3.ReportSource = cryRpt;

crystalReportViewer3.Refresh();

}

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

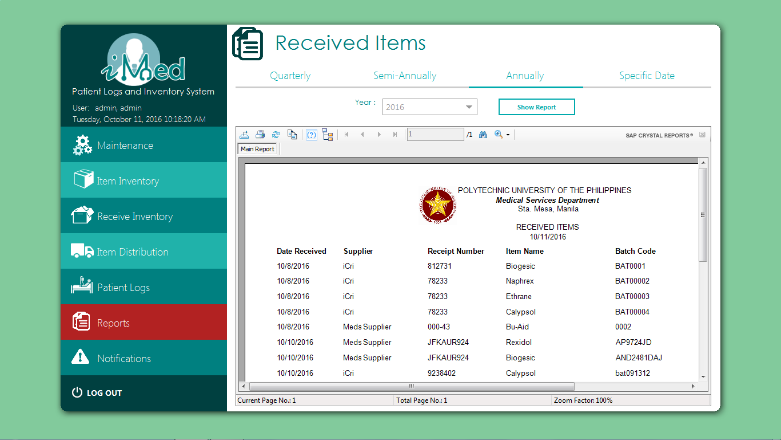
{

}

}

}

**Receive Item Form (Report)**

****

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using CrystalDecisions.Shared;

namespace PUPiMed.UserControls

{

public partial class UCReportReceive : UserControl

{

public UCReportReceive()

{

InitializeComponent();

int year = Convert.ToInt32(DateTime.Today.Year.ToString());

for (int x = year; x != year - 20; x--)

{

cbYear.Items.Add(x);

cbSemiYear.Items.Add(x);

cbAnnualYear.Items.Add(x);

}

cbYear.SelectedIndex = 0;

cbMonth.SelectedIndex = 0;

//quart();

cbSemiMonth.SelectedIndex = 0;

cbSemiYear.SelectedIndex = 0;

//semi();

cbAnnualYear.SelectedIndex = 0;

annual();

//specific();

dtTo.MaxDate = DateTime.Today;

}

public void quart()

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptReceive.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer1.ReportSource = cryRpt;

crystalReportViewer1.Refresh();

}

public void semi()

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptReceiveSemi.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbSemiMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbSemiYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer2.ReportSource = cryRpt;

crystalReportViewer2.Refresh();

}

public void annual()

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptReceiveAnnual.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbAnnualYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer3.ReportSource = cryRpt;

crystalReportViewer3.Refresh();

}

private void specific()

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptReceivePickADate.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = dtFrom.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["fromDate"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = dtTo.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["toDate"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer4.ReportSource = cryRpt;

crystalReportViewer4.Refresh();

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptReceive.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer1.ReportSource = cryRpt;

crystalReportViewer1.Refresh();

}

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

{

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptReceiveSemi.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbSemiMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbSemiYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer2.ReportSource = cryRpt;

crystalReportViewer2.Refresh();

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptReceiveAnnual.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbAnnualYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer3.ReportSource = cryRpt;

crystalReportViewer3.Refresh();

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptReceivePickADate.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = dtFrom.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["fromDate"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = dtTo.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["toDate"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer4.ReportSource = cryRpt;

crystalReportViewer4.Refresh();

}

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

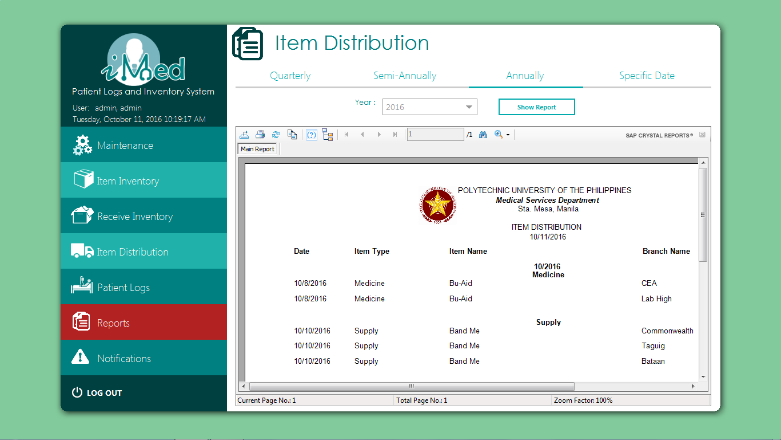
{

}

}

}

**Item Distribution (Report)**

****

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using CrystalDecisions.Shared;

namespace PUPiMed.UserControls

{

public partial class UCReportDistrib : UserControl

{

public UCReportDistrib()

{

InitializeComponent();

int year = Convert.ToInt32(DateTime.Today.Year.ToString());

for (int x = year; x != year - 20; x--)

{

cbYear.Items.Add(x);

cbSemiYear.Items.Add(x);

cbAnnualYear.Items.Add(x);

}

cbYear.SelectedIndex = 0;

cbMonth.SelectedIndex = 0;

//quart();

cbSemiMonth.SelectedIndex = 0;

cbSemiYear.SelectedIndex = 0;

//semi();

cbAnnualYear.SelectedIndex = 0;

//annual();

dtTo.MaxDate = DateTime.Today;

}

private void quart()

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDistribQuarterly.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer1.ReportSource = cryRpt;

crystalReportViewer1.Refresh();

}

private void semi()

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDistribSemi.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbSemiMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbSemiYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer2.ReportSource = cryRpt;

crystalReportViewer2.Refresh();

}

private void annual()

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDistribAnnual.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbAnnualYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer3.ReportSource = cryRpt;

crystalReportViewer3.Refresh();

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDistribPickADate.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = dtFrom.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["fromDate"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = dtTo.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["toDate"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer4.ReportSource = cryRpt;

crystalReportViewer4.Refresh();

}

private void metroButton1\_Click(object sender, EventArgs e)

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDistribQuarterly.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer1.ReportSource = cryRpt;

crystalReportViewer1.Refresh();

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptDistribSemi.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbSemiMonth.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["month"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = cbSemiYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer2.ReportSource = cryRpt;

crystalReportViewer2.Refresh();

}

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

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory+ @"\rptDistribAnnual.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = cbAnnualYear.Text;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["year"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer3.ReportSource = cryRpt;

crystalReportViewer3.Refresh();

}

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

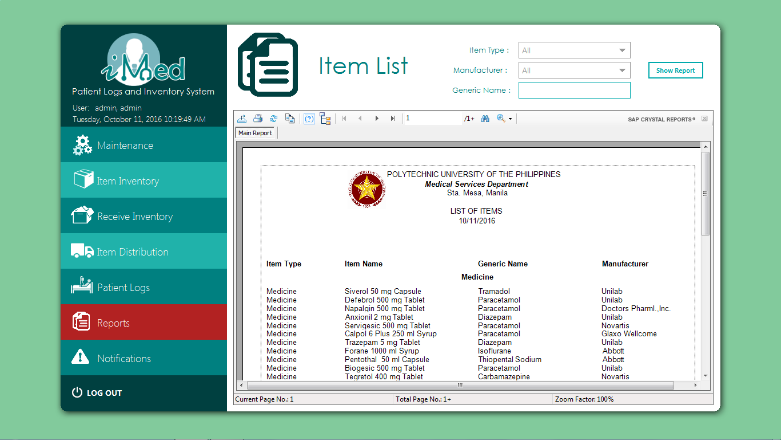
{

}

}

}

**List of Items (Report)**

****

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using CrystalDecisions.Shared;

using MySql.Data.MySqlClient;

using System.Data;

namespace PUPiMed.UserControls

{

public partial class UCReportItemList : UserControl

{

public UCReportItemList()

{

InitializeComponent();

String manu = "";

String gname = "";

String itemType = "";

//Rpt(manu, itemType, gname);

}

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

{

string itemType = "";

string manu = "";

string gname = txtGName.Text;

if (cbItemType.Text != "All")

{

itemType = cbItemType.Text;

}

if (cbManufacturer.Text != "All")

{

manu = cbManufacturer.Text;

}

Rpt(manu,itemType,gname);

}

public void Rpt(String manu, String itemType, String gname)

{

ReportDocument cryRpt = new ReportDocument();

cryRpt.Load(AppDomain.CurrentDomain.BaseDirectory + @"\rptItemList1.rpt");

TableLogOnInfos crtableLogoninfos = new TableLogOnInfos();

TableLogOnInfo crtableLogoninfo = new TableLogOnInfo();

ConnectionInfo crConnectionInfo = new ConnectionInfo();

Tables CrTables;

ParameterFieldDefinitions crParameterFieldDefinitions;

ParameterFieldDefinition crParameterFieldDefinition;

ParameterValues crParameterValues = new ParameterValues();

ParameterDiscreteValue crParameterDiscreteValue = new ParameterDiscreteValue();

crParameterDiscreteValue.Value = gname;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["genericName"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = itemType;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["itemType"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

crParameterDiscreteValue.Value = manu;

crParameterFieldDefinitions = cryRpt.DataDefinition.ParameterFields;

crParameterFieldDefinition = crParameterFieldDefinitions["manufacturer"];

crParameterValues = crParameterFieldDefinition.CurrentValues;

crParameterValues.Clear();

crParameterValues.Add(crParameterDiscreteValue);

crParameterFieldDefinition.ApplyCurrentValues(crParameterValues);

SqlConnectionStringBuilder SqlConnectionStringBuilder = new SqlConnectionStringBuilder(LoginNew.conn.ConnectionString);

string ServerName = SqlConnectionStringBuilder.DataSource;

string DatabaseName = SqlConnectionStringBuilder.InitialCatalog;

Boolean IntegratedSecurity = SqlConnectionStringBuilder.IntegratedSecurity;

string UserID = SqlConnectionStringBuilder.UserID;

string Password = SqlConnectionStringBuilder.Password;

foreach (CrystalDecisions.CrystalReports.Engine.Table Table in cryRpt.Database.Tables)

{

TableLogOnInfo TableLogOnInfo = Table.LogOnInfo;

TableLogOnInfo.ConnectionInfo.ServerName = ServerName;

TableLogOnInfo.ConnectionInfo.DatabaseName = DatabaseName;

TableLogOnInfo.ConnectionInfo.IntegratedSecurity = IntegratedSecurity;

if (IntegratedSecurity != true)

{

TableLogOnInfo.ConnectionInfo.UserID = UserID;

TableLogOnInfo.ConnectionInfo.Password = Password;

}

Table.ApplyLogOnInfo(TableLogOnInfo);

}

crystalReportViewer1.ReportSource = cryRpt;

crystalReportViewer1.Refresh();

}

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

{

txtGName.Clear();

}

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

{

using (LoginNew.conn)

{

if (LoginNew.conn != null && (LoginNew.conn.State == ConnectionState.Closed))

LoginNew.conn.Open();

using (MySqlCommand cmd = new MySqlCommand("SELECT strManuName FROM tblManufacturer WHERE boolManuDeleted=FALSE;", LoginNew.conn))

{

using (MySqlDataReader dataReader = cmd.ExecuteReader())

{

if (dataReader.Read())

{

cbManufacturer.Items.Add(dataReader.GetString(0));

while (dataReader.Read())

{

cbManufacturer.Items.Add(dataReader.GetString(0));

}

}

else

{

cbManufacturer.Items.Add("");

}

dataReader.Close();

cbManufacturer.Selec­tedIndex = 0;

}

}

}

cbItemType.SelectedIndex = 0;

}

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

{

}

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

{

}

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

{

}

}

}

**Notifications Form**

****

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace ToastNotifications

{

public partial class Notification : Form

{

private static readonly List<Notification> OpenNotifications = new List<Notification>();

private bool \_allowFocus;

private readonly FormAnimator \_animator;

private IntPtr \_currentForegroundWindow;

/// <summary>

///

/// </summary>

/// <param name="title"></param>

/// <param name="body"></param>

/// <param name="duration"></param>

/// <param name="animation"></param>

/// <param name="direction"></param>

public Notification(string title, string body, int duration, FormAnimator.AnimationMethod animation, FormAnimator.AnimationDirection direction)

{

InitializeComponent();

if (duration < 0)

duration = int.MaxValue;

else

duration = duration \* 1000;

lifeTimer.Interval = duration;

labelTitle.Text = title;

labelBody.Text = body;

\_animator = new FormAnimator(this, animation, direction, 500);

Region = Region.FromHrgn(NativeMethods.CreateRoundRectRgn(0, 0, Width - 5, Height - 5, 20, 20));

}

#region Methods

/// <summary>

/// Displays the form

/// </summary>

/// <remarks>

/// Required to allow the form to determine the current foreground window before being displayed

/// </remarks>

public new void Show()

{

// Determine the current foreground window so it can be reactivated each time this form tries to get the focus

\_currentForegroundWindow = NativeMethods.GetForegroundWindow();

base.Show();

}

#endregion // Methods

#region Event Handlers

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

{

// Display the form just above the system tray.

Location = new Point(Screen.PrimaryScreen.WorkingArea.Width - Width,

Screen.PrimaryScreen.WorkingArea.Height - Height);

// Move each open form upwards to make room for this one

foreach (Notification openForm in OpenNotifications)

{

openForm.Top -= Height;

}

OpenNotifications.Add(this);

lifeTimer.Start();

}

private void Notification\_Activated(object sender, EventArgs e)

{

// Prevent the form taking focus when it is initially shown

if (!\_allowFocus)

{

// Activate the window that previously had focus

NativeMethods.SetForegroundWindow(\_currentForegroundWindow);

}

}

private void Notification\_Shown(object sender, EventArgs e)

{

// Once the animation has completed the form can receive focus

\_allowFocus = true;

// Close the form by sliding down.

\_animator.Duration = 0;

\_animator.Direction = FormAnimator.AnimationDirection.Down;

}

private void Notification\_FormClosed(object sender, FormClosedEventArgs e)

{

// Move down any open forms above this one

foreach (Notification openForm in OpenNotifications)

{

if (openForm == this)

{

// Remaining forms are below this one

break;

}

openForm.Top += Height;

}

OpenNotifications.Remove(this);

}

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

{

Close();

}

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

{

Close();

}

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

{

Close();

}

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

{

Close();

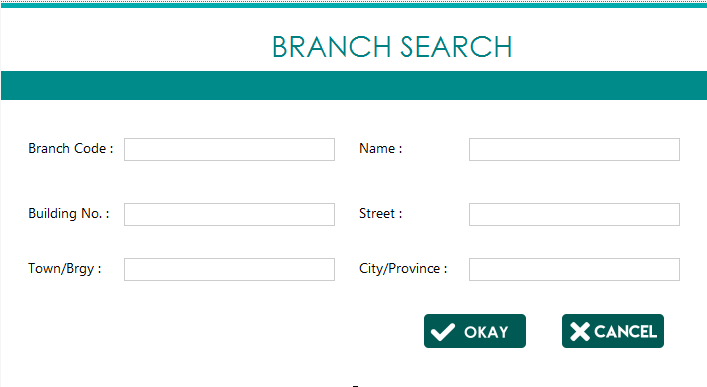
}

#endregion // Event Handlers

}

}

**Branch Search**



using MetroFramework.Forms;

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;

using PUPiMed.UserControls;

namespace PUPiMed

{

public partial class FormBranchSearch : MetroForm

{

private UCBranches uCBranches;

public FormBranchSearch()

{

InitializeComponent();

}

public FormBranchSearch(UCBranches uCBranches)

{

InitializeComponent();

this.uCBranches = uCBranches;

}

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

{

this.Dispose();

}

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

{

this.uCBranches.query =

"SELECT "+

" a.strBranCode AS `Code`, " +

" a.strBranDesc AS `Branch Name`, " +

" CONCAT(b.strAddrNum, " +

" ' ', " +

" b.strAddrStreet, " +

" ', ', " +

" b.strAddrTown, " +

" ', ', " +

" b.strAddrCity) AS `Address` " +

"FROM " +

" (tblbranch a" +

" LEFT JOIN tbladdress b ON((a.strBranAddrCode = b.strAddrCode))) " +

"WHERE " +

" a.boolBranDeleted = 0 AND" +

" a.strBranCode LIKE '"+txtCode.Text+"%' AND" +

" a.strBranDesc LIKE '" + txtName.Text + "%' AND" +

" b.strAddrNum LIKE '" + txtNo.Text + "%' AND" +

" b.strAddrStreet LIKE '" + txtStreet.Text + "%' AND" +

" b.strAddrTown LIKE '" + txtTown.Text + "%' AND" +

" b.strAddrCity LIKE '" + txtCity.Text + "%'" +

" ;";

this.uCBranches.updateTable();

this.Dispose();

}

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

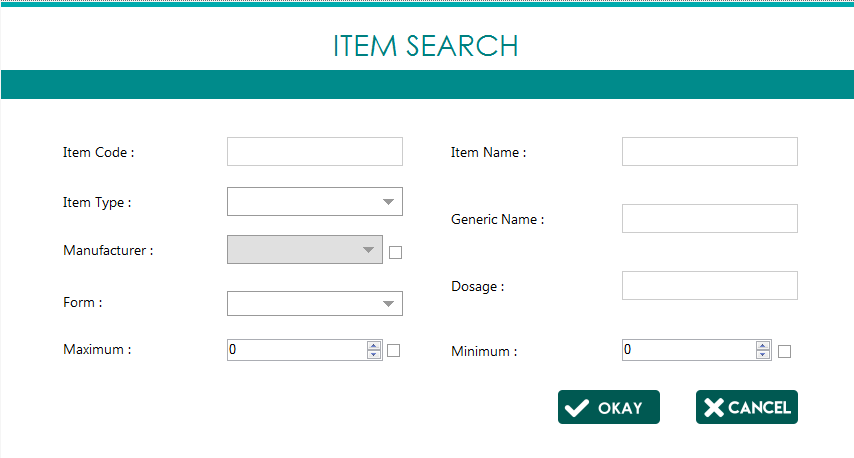
{

}

}

}

**Item Search**



using MetroFramework.Forms;

using System;

using PUPiMed.UserControls;

using System.Collections.Generic;

using System.Windows.Forms;

using System.Drawing;

namespace PUPiMed

{

public partial class FormItemSearch : MetroForm

{

private int itemType = 0;

private UCItem uCItem;

private UCItemDistribution uCItemDistribution;

private UCItemInventory uCItemInventory;

List<String> listManuCode;

string strType = "";

public FormItemSearch()

{

InitializeComponent();

loadManufacturer();

cbForm.SelectedIndex = 0;

}

public FormItemSearch(UCItemDistribution uCItemDistribution) : this()

{

this.uCItemDistribution = uCItemDistribution;

}

public FormItemSearch(UCItem uCItem, int itemType) : this()

{

this.uCItem = uCItem;

this.itemType = itemType;

this.cbType.SelectedIndex = itemType;

this.cbType.Enabled = false;

}

public FormItemSearch(UCItemInventory uCItemInventory) : this()

{

this.uCItemInventory = uCItemInventory;

}

private void metroLabel5\_EnabledChanged(object sender, EventArgs e)

{

(sender as Control).BackColor = (sender as Control).Enabled? Color.White : Color.FromArgb(224,224,224);

}

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

{

cbManufacturer.Enabled = chkManu.Checked;

}

public void loadManufacturer()

{

string strManu = "";

switch (cbType.SelectedIndex)

{

case 0:

strManu = "";

break;

case 1:

strManu = " AND intManuType=1 ";

break;

case 2:

strManu = " AND intManuType=2 ";

break;

case 3:

strManu = " AND intManuType=3 ";

break;

}

if (Program.fillComboBox(cbManufacturer, "SELECT \* FROM tblManufacturer WHERE boolManuDeleted = false "+strManu+";", out listManuCode))

{

cbManufacturer.PromptText = "Please add a new Manufacturer";

}

}

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

{

string query =

" AND Code LIKE '" + txtCode.Text + "%' AND Code in (select strItemCode from tblItem WHERE " +

strType +

" AND strItemName LIKE '" + txtName.Text + "%' " +

" AND strItemGeneric LIKE '" + txtGen.Text + "%' " +

(chkManu.Checked ? (" AND strItemManuCode LIKE '" + listManuCode[cbManufacturer.SelectedIndex].ToString() + "%' ") : "") +

(txtDosage.Enabled ? (" AND strItemCode IN (SELECT strAddiItemCode FROM tbladditional WHERE strAddiDosage LIKE '" + txtDosage.Text + "%') ") : "") +

(cbForm.Enabled ? (" AND strItemCode IN (SELECT strAddiItemCode FROM tbladditional WHERE strAddiForm LIKE '" + cbForm.Text + "%') ") : "") +

(chkMin.Checked ? (" AND intItemMin = " + txtMin.Text) : "") +

(chkMax.Checked ? (" AND intItemMax = " + txtMax.Text) : "") +

"); ";

if (uCItem != null)

{

this.uCItem.query += query;

this.uCItem.updateTable();

this.Dispose();

}

if (uCItemDistribution != null)

{

this.uCItemDistribution.query += query;

this.uCItemDistribution.updateTable();

this.Dispose();

}

if(uCItemInventory != null)

{

this.uCItemInventory.query += query;

this.uCItemInventory.updateTable(this.uCItemInventory.query, this.uCItemInventory.grid);

this.Dispose();

}

}

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

{

itemType = cbType.SelectedIndex;

switch (cbType.SelectedIndex)

{

case 0:

strType = "";

break;

case 1:

strType = " intItemType=1 ";

txtDosage.Enabled = true;

cbForm.Enabled = true;

loadManufacturer();

break;

case 2:

strType = " intItemType=2 ";

txtDosage.Enabled = true;

cbForm.Enabled = true;

loadManufacturer();

break;

case 3:

strType = " intItemType=3 ";

txtDosage.Enabled = false;

cbForm.Enabled = false;

loadManufacturer();

break;

}

}

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

{

txtMin.Enabled = chkMin.Checked;

}

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

{

txtMax.Enabled = chkMax.Checked;

}

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

{

this.Dispose();

}

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

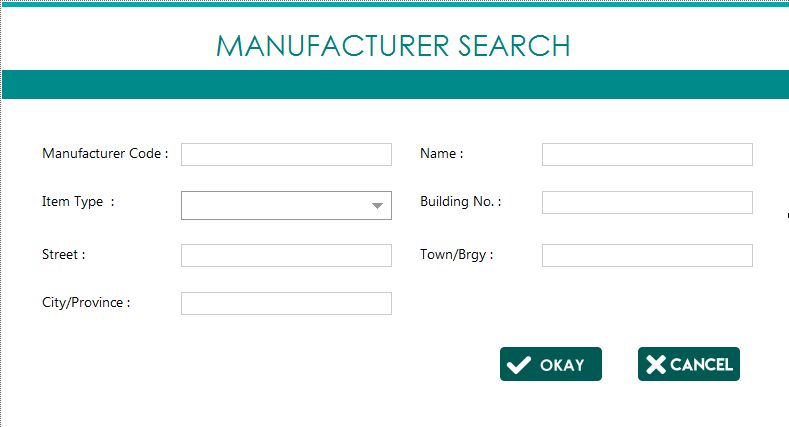
{

}

}

}

**Manufacturer Search**



using MetroFramework.Forms;

using System;

using PUPiMed.UserControls;

namespace PUPiMed

{

public partial class FormManufacturerSearch : MetroForm

{

private UCManufacturer uCManufacturer;

public FormManufacturerSearch()

{

InitializeComponent();

}

public FormManufacturerSearch(UCManufacturer uCManufacturer)

{

InitializeComponent();

this.uCManufacturer = uCManufacturer;

}

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

{

}

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

{

string strType ="";

switch (cbItemType.SelectedIndex)

{

case 0:

strType = "";

break;

case 1:

strType = " intManuType=1 AND";

break;

case 2:

strType = " intManuType=2 AND";

break;

case 3:

strType = " intManuType=3 AND";

break;

}

this.uCManufacturer.query =

"SELECT " +

" a.strManuCode AS `Code`, " +

" a.strManuName AS `Manufacturer's Name`, " +

" (CASE " +

" WHEN(`a`.`intManuType` = 1) THEN 'Medicine' " +

" WHEN(`a`.`intManuType` = 2) THEN 'Supply' " +

" WHEN(`a`.`intManuType` = 3) THEN 'Equipment' " +

" END) AS `Item Type`, " +

" CONCAT(b.strAddrNum, " +

" ' ', " +

" b.strAddrStreet, " +

" ', ', " +

" b.strAddrTown, " +

" ', ', " +

" b.strAddrCity) AS `Address` " +

"FROM " +

" tblmanufacturer a" +

" LEFT JOIN tbladdress b ON a.strManuAddrCode = b.strAddrCode " +

"WHERE " +

" a.boolManuDeleted = 0 AND" +

" a.strManuCode LIKE '" + txtCode.Text + "%' AND" +

strType +

" a.strManuName LIKE '" + txtName.Text + "%' AND" +

" b.strAddrNum LIKE '" + txtNo.Text + "%' AND" +

" b.strAddrStreet LIKE '" + txtStreet.Text + "%' AND" +

" b.strAddrTown LIKE '" + txtTown.Text + "%' AND" +

" b.strAddrCity LIKE '" + txtCity.Text + "%'" +

" ;";

this.uCManufacturer.updateTable();

this.Dispose();

}

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

{

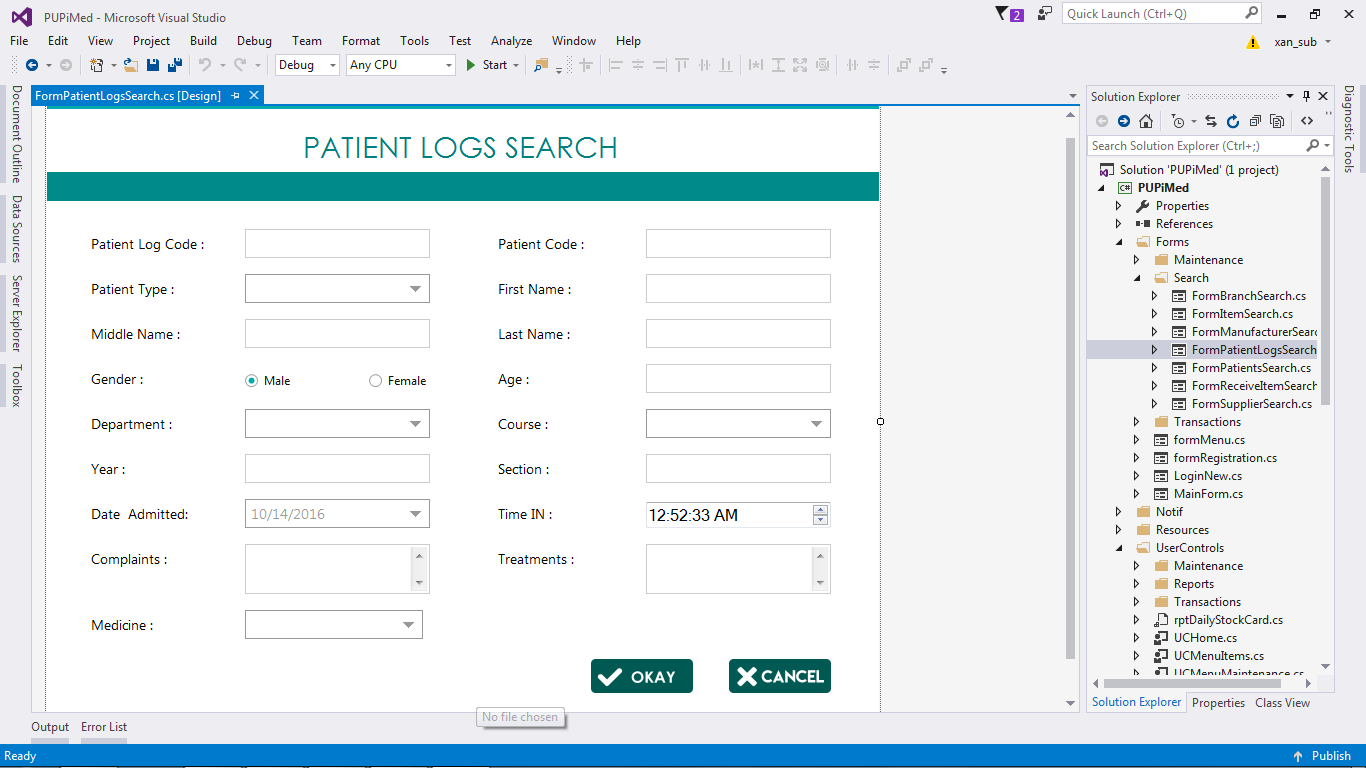
this.Dispose();

}

}

}

**Patient Logs Search**



using MetroFramework.Forms;

using System;

namespace PUPiMed.Forms

{

public partial class FormPatientLogsSearch : MetroForm

{

public FormPatientLogsSearch()

{

InitializeComponent();

}

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

{

}

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

{

this.Dispose();

}

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

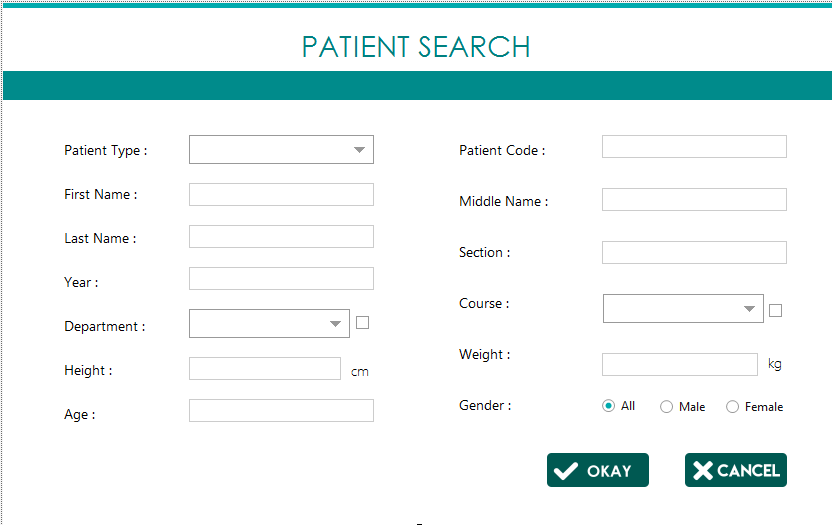
{

}

}

}

**Patient Search**



using MetroFramework.Forms;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace PUPiMed

{

public partial class FormPatientSearch : MetroForm

{

string query = "", strCollege = "", strDept = "", strType, strGender = "";

List<String> listDept, listCourse;

private UCPatientLogs\_DTR\_ uCPatientLogs\_DTR\_;

private UCStudentTab uCStudentTab;

private int intType = 0;

private UCFacultyTab uCFacultyTab;

private UCAdminTab uCAdminTab;

private FormAddPatientLogs formAddPatientLogs;

public FormPatientSearch()

{

InitializeComponent();

}

public FormPatientSearch(UCPatientLogs\_DTR\_ uCPatientLogs\_DTR\_) : this()

{

this.uCPatientLogs\_DTR\_ = uCPatientLogs\_DTR\_;

query = "SELECT `Date`, `Time In`, `First Name`, `Middle Name`, `Last Name`, `Department`, `Complaints` FROM vwlogs WHERE ";

}

public FormPatientSearch(UCStudentTab uCStudentTab) : this()

{

this.uCStudentTab = uCStudentTab;

this.intType = 1;

this.cbPatientType.SelectedIndex = intType;

this.cbPatientType.Enabled = false;

loadType();

query = "SELECT \* FROM vwstudentlogs WHERE ";

}

public FormPatientSearch(UCFacultyTab uCFacultyTab) : this()

{

this.uCFacultyTab = uCFacultyTab;

this.intType = 2;

this.cbPatientType.SelectedIndex = intType;

this.cbPatientType.Enabled = false;

loadType();

query = "SELECT \* FROM vwfacultylogs WHERE ";

}

public FormPatientSearch(UCAdminTab uCAdminTab) : this()

{

this.uCAdminTab = uCAdminTab;

this.intType = 3;

this.cbPatientType.SelectedIndex = intType;

this.cbPatientType.Enabled = false;

loadType();

query = "SELECT \* FROM vwadminlogs WHERE ";

}

public FormPatientSearch(FormAddPatientLogs formAddPatientLogs) : this()

{

this.formAddPatientLogs = formAddPatientLogs;

query = "SELECT Code, `Name` from vwpatients WHERE ";

}

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

{

}

private void loadType()

{

if (intType <= 1)

{

cbCourse.Enabled = true;

txtYear.Enabled = true;

txtSection.Enabled = true;

lblDept.Text = "College :";

loadDepartment();

loadCourse();

}

else if (intType == 2)

{

cbCourse.Enabled = false;

txtYear.Enabled = false;

txtSection.Enabled = false;

lblDept.Text = "College :";

loadDepartment();

}

else

{

cbCourse.Enabled = false;

txtYear.Enabled = false;

txtSection.Enabled = false;

lblDept.Text = "Department :";

loadDepartment();

}

}

private void loadDepartment()

{

if (chkDept.Checked)

{

string strTable = "";

if (intType == 3)

strTable = "SELECT \* FROM tbldepartment WHERE boolDeptDeleted = FALSE;";

else

strTable = "SELECT \* FROM tblcollege WHERE boolCollDeleted = FALSE;";

if (!Program.fillComboBox(cbDept, strTable, out listDept))

{

cbDept.PromptText = "Nothing Found!";

}

cbDept.SelectedIndex = 0;

}

}

private void loadCourse()

{

if (chkCourse.Checked)

{

if (cbDept.Items.Count != 0)

{

if (!Program.fillComboBox(cbCourse, "SELECT \* FROM tblCourse WHERE strCourCollCode = '" + listDept[cbDept.SelectedIndex].ToString() + "';", out listCourse))

{

cbCourse.PromptText = "No Course Found!";

}

}

}

}

private void txtAge\_EnabledChanged(object sender, EventArgs e)

{

(sender as Control).BackColor = (sender as Control).Enabled ? Color.White : Color.FromArgb(224, 224, 224);

}

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

{

cbDept.Enabled = chkDept.Checked;

if (cbDept.Enabled)

{

loadDepartment();

}

}

private void chkCourse\_CheckedChanged\_1(object sender, EventArgs e)

{

cbCourse.Enabled = chkCourse.Checked;

if (cbCourse.Enabled)

{

loadCourse();

}

}

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

{

if (cbPatientType.SelectedIndex == 0)

{

loadCourse();

}

else

{

cbCourse.PromptText = "";

}

}

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

{

this.Dispose();

}

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

{

if (intType <= 1)

{

if(chkCourse.Checked)

strCollege = (chkCourse.Checked==false ? "" : (" AND strPatiCode IN (SELECT strStuiCourCode FROM tblstudentinfo WHERE strStuiCourCode LIKE '" + listCourse[cbCourse.SelectedIndex].ToString() + "%') "));

strCollege+=(string.IsNullOrEmpty(txtYear.Text) ? "" : (" AND strPatiCode IN(SELECT strStuiPatiCode FROM tblstudentinfo WHERE strStuiYear LIKE '" + txtYear.Text + "%') "));

strCollege+=(string.IsNullOrEmpty(txtSection.Text) ? "" : (" AND strPatiCode IN(SELECT strStuiPatiCode FROM tblstudentinfo WHERE strStuiSection LIKE '" + txtSection.Text + "%') "));

strDept = "";

}

else

{

strCollege = "";

if(chkDept.Checked)

if(intType==2)

strDept = (chkDept.Checked==false ? "" : (" AND strPatiCode IN (SELECT strFacIPatiCode FROM tblfacultyinfo WHERE strFacICollCode LIKE '" + listDept[cbDept.SelectedIndex].ToString() + "%') "));

else

strDept = (chkDept.Checked==false ? "" : (" AND strPatiCode IN (SELECT strAdmIPatiCode FROM tbladmininfo WHERE strAdmIDeptCode LIKE '" + listDept[cbDept.SelectedIndex].ToString() + "%') "));

}

if (rbAll.Checked)

strGender = "";

else if (rbMale.Checked)

strGender = " AND intPatiGender = 0 ";

else

strGender = " AND intPatiGender = 1 ";

query +=

" `Code` LIKE '" + txtCode.Text + "%' AND `Code` IN (" +

" SELECT strPatiCode FROM dbmedicalclinic.tblpatient " +

" WHERE strPatiFirstName LIKE '" + txtFName.Text + "%' " +

" AND strPatiMiddleName LIKE '" + txtMName.Text + "%' " +

" AND strPatiLastName LIKE '" + txtLName.Text + "%' " +

(string.IsNullOrEmpty(txtHeight.Text) ? "" : " AND fltPatiHeight = " + txtHeight.Text) +

(string.IsNullOrEmpty(txtWeight.Text) ? "" : " AND fltPatiWeight = " + txtWeight.Text) +

(string.IsNullOrEmpty(txtAge.Text) ? "" : " AND TIMESTAMPDIFF(YEAR, datPatiDOB, CURDATE()) = " +txtAge.Text) +

strGender + strType + strCollege + strDept +

" ); ";

//MessageBox.Show(query);

if (uCStudentTab != null)

{

this.uCStudentTab.strQuery = query;

this.uCStudentTab.updateTable();

}

if(uCPatientLogs\_DTR\_ != null)

{

this.uCPatientLogs\_DTR\_.strQuery = query;

this.uCPatientLogs\_DTR\_.updateTable();

}

if (uCFacultyTab != null)

{

this.uCFacultyTab.strQuery = query;

this.uCFacultyTab.updateTable();

}

if (uCAdminTab!= null)

{

this.uCAdminTab.strQuery = query;

this.uCAdminTab.updateTable();

}

if (formAddPatientLogs != null)

{

formAddPatientLogs.loadPatients(query);

}

this.Dispose();

}

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

{

this.intType = cbPatientType.SelectedIndex;

loadType();

switch (cbPatientType.SelectedIndex)

{

case 0:

strType = "";

break;

case 1:

strType = " AND intPatiType = 0 ";

break;

case 2:

strType = " AND intPatiType = 1 ";

break;

case 3:

strType = " AND intPatiType = 2 ";

break;

}

}

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

{

}

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

{

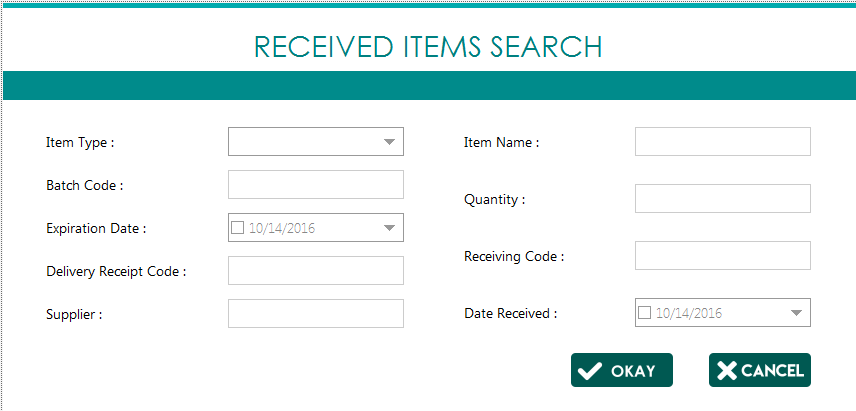
e.Handled = !(char.IsNumber(e.KeyChar)) && !(char.IsControl(e.KeyChar));

}

}

}

**Received Item Search**

using MetroFramework.Forms;

namespace PUPiMed.Forms

{

public partial class FormReceiveItemSearch : MetroForm

{

private UCReceive uCReceive;

string strType;

public FormReceiveItemSearch()

{

InitializeComponent();

}

private void cbType\_SelectedIndexChanged(object sender, System.EventArgs e)

{

switch (cbType.SelectedIndex)

{

case 0:

strType = "";

break;

default:

strType = "c.intItemType = "+cbType.SelectedIndex+" AND ";

break;

}

}

public FormReceiveItemSearch(UCReceive uCReceive) : this()

{

this.uCReceive = uCReceive;

}

private void btncancel\_Click(object sender, System.EventArgs e)

{

this.Dispose();

}

private void FormReceiveItemSearch\_Load(object sender, System.EventArgs e)

{

}

private void btnAdd\_Click(object sender, System.EventArgs e)

{

string query =

"SELECT " +

" `a`.`strRecHCode` AS `Receiving Code`, " +

" `c`.`strItemName` AS `Item Name`, " +

" `b`.`intRecDQty` AS `Quantity`, " +

" `b`.`datRecDExpDate` AS `Expiration Date`, " +

" `d`.`strSuppName` AS `Supplier`, " +

" `a`.`datRecHDateReceived` AS `Date Received` " +

"FROM " +

" `dbmedicalclinic`.`tblreceiveheader` `a` " +

" LEFT JOIN `dbmedicalclinic`.`tblreceivedetail` `b` ON `a`.`strRecHCode` = `b`.`strRecDRecHCode` " +

" JOIN `dbmedicalclinic`.`tblitem` `c` ON `b`.`strRecDItemCode` = `c`.`strItemCode` " +

" JOIN `dbmedicalclinic`.`tblsupplier` `d` ON `a`.`strRecHSuppCode` = `d`.`strSuppCode` " +

"WHERE " +

" `c`.`boolItemDeleted` <> 1 AND " +

strType +

(string.IsNullOrEmpty(txtQty.Text) ? "" :(" b.intRecDQty = " + txtQty.Text + " AND ")) +

(dtReceived.Checked?(" a.datRecHDateReceived = '" + dtReceived.Value.ToString("yyyy-MM-dd") + "' AND "): "") +

(dtExp.Checked ?(" b.datRecDExpDate = '" + dtExp.Value.ToString("yyyy-MM-dd") + "' AND ") : "") +

" c.strItemName LIKE '" + txtName.Text + "%' AND " +

" d.strSuppName LIKE '" + txtSupplier.Text + "%' AND " +

" b.strRecDBatchCode LIKE '" + txtBatch.Text + "%' AND " +

" a.strRecHCode LIKE '" + txtRCode.Text + "%' AND " +

" a.strRecHDeliveryReceipt LIKE '" + txtReceipt.Text + "%' " +

"ORDER BY `a`.`strRecHCode`; ";

this.uCReceive.strQuery = query;

this.uCReceive.updateTable();

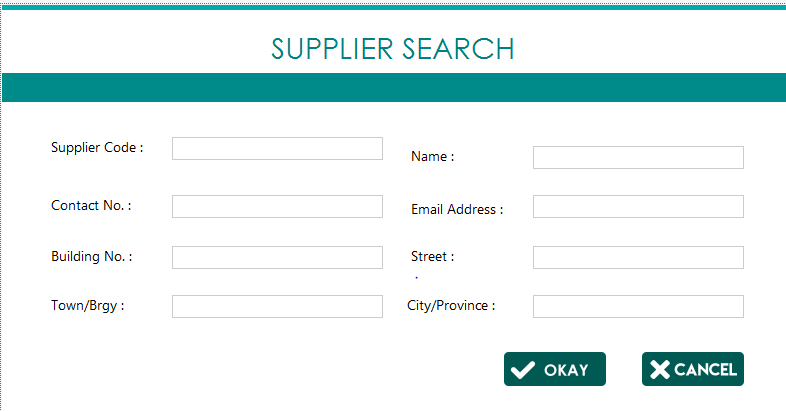
this.Dispose();

}

}

}

**Supplier Search**



using PUPiMed.UserControls;

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 PUPiMed

{

public partial class FormSupplierSearch : MetroFramework.Forms.MetroForm

{

public UCSupplier ucSupplier;

public FormSupplierSearch()

{

InitializeComponent();

}

public FormSupplierSearch(UCSupplier ucSupplier)

{

InitializeComponent();

this.ucSupplier = ucSupplier;

}

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

{

this.Dispose();

}

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

{

this.ucSupplier.query =

"SELECT " +

" a.strSuppCode AS `Code`, " +

" a.strSuppName AS `Supplier's Name`, " +

" CONCAT(b.strAddrNum, " +

" ' ', " +

" b.strAddrStreet, " +

" ', ', " +

" b.strAddrTown, " +

" ', ', " +

" b.strAddrCity) AS `Address`, " +

" a.strSuppEmail AS `Email`, " +

" a.strSuppContactNo AS `Contact No.` "+

"FROM " +

" (tblsupplier a" +

" LEFT JOIN tbladdress b ON((a.strSuppAddrCode = b.strAddrCode))) " +

"WHERE " +

" a.boolSuppDeleted = 0 AND" +

" a.strSuppCode LIKE '" + txtCode.Text + "%' AND" +

" a.strSuppName LIKE '" + txtName.Text + "%' AND" +

" a.strSuppEmail LIKE '" + txtEmail.Text +"%' AND" +

" a.strSuppContactNo LIKE '" + txtContact.Text + "%' AND" +

" b.strAddrNum LIKE '" + txtNo.Text + "%' AND" +

" b.strAddrStreet LIKE '" + txtStreet.Text + "%' AND" +

" b.strAddrTown LIKE '" + txtTown.Text + "%' AND" +

" b.strAddrCity LIKE '" + txtCity.Text + "%'" +

" ;";

this.ucSupplier.updateTable();

this.Dispose();

}

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

{

}

}

}