using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace FailurePredictionArtificialLiftSystem

{

public partial class FrmPumpRegistration : Form

{

private OleDbConnection cn = new OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + Application.StartupPath + "\\DbArtificialLift.accdb;Persist Security Info=True");

private OleDbCommand cmd = new OleDbCommand();

public FrmPumpRegistration()

{

InitializeComponent();

}

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

{

this.Hide();

}

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

{

cmd.Connection = cn;

}

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

{

try

{

if (isAllEntryFilled())

{

string PumpID = txtPumpID.Text;

string ArtificialLift = cboArtificialLift.Text;

string Pump = cboPump.Text;

string OperatingRange = cboOperatingRange.Text;

string Motor = cboMotor.Text;

string Cable = cboCable.Text;

int PumpDepth = Convert.ToInt32(txtPumpDepth.Text);

int OperatingFrequency = Convert.ToInt16(txtOperatingFrequency.Text);

{

cn.Open();

cmd.CommandText = "insert into tblPumpReg values('" + PumpID + "','" + ArtificialLift + "','" + Pump + "','" + OperatingRange + "','" + Motor + "','" + Cable + "'," + PumpDepth + "," + OperatingFrequency + ")";

cmd.ExecuteNonQuery();

MessageBox.Show("New Pump Registration Details Successfully Saved!!!", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Information);

cn.Close();

btnRefresh();

}

}

}

catch (Exception er)

{

cn.Close();

MessageBox.Show(er.Message, "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private bool isAllEntryFilled()

{

bool isEntryFilled = false;

if ((txtPumpID.Text == ""))

{

MessageBox.Show("Please type the Pump ID", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtPumpID.Focus();

return isEntryFilled;

}

else if ((cboArtificialLift.Text == ""))

{

MessageBox.Show("Please type the Artificial Lift", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboArtificialLift.Focus();

return isEntryFilled;

}

else if ((cboPump.Text == ""))

{

MessageBox.Show("Please set the Pump Type", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboPump.Focus();

return isEntryFilled;

}

else if ((cboOperatingRange.Text == ""))

{

MessageBox.Show("Please set the Operating Range", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboOperatingRange.Focus();

return isEntryFilled;

}

else if ((cboMotor.Text == ""))

{

MessageBox.Show("Please set the Pump Motor", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboMotor.Focus();

return isEntryFilled;

}

else if ((cboCable.Text == ""))

{

MessageBox.Show("Please type the Cable ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboCable.Focus();

return isEntryFilled;

}

else if ((txtPumpDepth.Text == ""))

{

MessageBox.Show("Please type the Pump Depth ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtPumpDepth.Focus();

return isEntryFilled;

}

else if ((txtOperatingFrequency.Text == ""))

{

MessageBox.Show("Please type the Operating Frequency ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtOperatingFrequency.Focus();

return isEntryFilled;

}

else

return true;

}

private void btnRefresh()

{

txtPumpID.Clear();

cboArtificialLift.ResetText();

cboPump.ResetText();

cboOperatingRange.ResetText();

cboMotor.ResetText();

cboCable.ResetText();

txtPumpDepth.ResetText();

txtOperatingFrequency.ResetText();

txtPumpID.Focus();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace FailurePredictionArtificialLiftSystem

{

public partial class FrmWellRegistration : Form

{

private OleDbConnection cn = new OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + Application.StartupPath + "\\DbArtificialLift.accdb;Persist Security Info=True");

private OleDbCommand cmd = new OleDbCommand();

public FrmWellRegistration()

{

InitializeComponent();

}

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

{

this.Hide();

}

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

{

try

{

if (isAllEntryFilled())

{

string WellID = txtWellID.Text;

string WellPermit = txtWellPermit.Text;

string OperatorName = txtOperatorName.Text;

string Field = txtField.Text;

string Country = cboCountry.Text;

string Location = cboLocation.Text;

string WellName = txtWellName.Text;

string WellStatus = cboWellStatus.Text;

string WellType = cboWellType.Text;

string FlowType = cboFlowType.Text;

{

cn.Open();

cmd.CommandText = "insert into tblWellReg values('" + WellID + "','" + WellPermit + "','" + OperatorName + "','" + Field + "','" + Country + "','" + Location + "','" + WellName + "','" + WellStatus + "','" + WellType + "','" + FlowType + "')";

cmd.ExecuteNonQuery();

MessageBox.Show("New Well Registration Details Successfully Saved!!!", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Information);

cn.Close();

btnRefresh();

}

}

}

catch (Exception er)

{

cn.Close();

MessageBox.Show(er.Message, "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private bool isAllEntryFilled()

{

bool isEntryFilled = false;

if ((txtWellID.Text == ""))

{

MessageBox.Show("Please type the Well ID", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtWellID.Focus();

return isEntryFilled;

}

else if ((txtWellPermit.Text == ""))

{

MessageBox.Show("Please type the Well Permit", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtWellPermit.Focus();

return isEntryFilled;

}

else if ((txtOperatorName.Text == ""))

{

MessageBox.Show("Please set the Operator Name", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtOperatorName.Focus();

return isEntryFilled;

}

else if ((txtField.Text == ""))

{

MessageBox.Show("Please set the Field", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtField.Focus();

return isEntryFilled;

}

else if ((cboCountry.Text == ""))

{

MessageBox.Show("Please set the Country", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboCountry.Focus();

return isEntryFilled;

}

else if ((cboLocation.Text == ""))

{

MessageBox.Show("Please type the Location ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboLocation.Focus();

return isEntryFilled;

}

else if ((txtWellName.Text == ""))

{

MessageBox.Show("Please type the Well Name ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtWellName.Focus();

return isEntryFilled;

}

else if ((cboWellStatus.Text == ""))

{

MessageBox.Show("Please type the Well Status ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboWellStatus.Focus();

return isEntryFilled;

}

else if ((cboWellType.Text == ""))

{

MessageBox.Show("Please type the Well Type ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboWellType.Focus();

return isEntryFilled;

}

else if ((cboFlowType.Text == ""))

{

MessageBox.Show("Please type the Flow Type ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboFlowType.Focus();

return isEntryFilled;

}

else

return true;

}

private void btnRefresh()

{

txtWellID.Clear();

txtWellPermit.Clear();

txtOperatorName.ResetText();

txtField.ResetText();

cboCountry.ResetText();

cboLocation.ResetText();

txtWellName.ResetText();

cboWellStatus.ResetText();

cboWellType.ResetText();

cboFlowType.ResetText();

}

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

{

cmd.Connection = cn;

}

private void label11\_Click(object sender, EventArgs e)

{

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace FailurePredictionArtificialLiftSystem

{

public partial class FrmDailyMeasure : Form

{

private OleDbConnection cn = new OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + Application.StartupPath + "\\DbArtificialLift.accdb;Persist Security Info=True");

private OleDbCommand cmd = new OleDbCommand();

private OleDbDataReader dr;

private bool nonNumberEntered;

public FrmDailyMeasure()

{

InitializeComponent();

}

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

{

cmd.Connection = cn;

loadWellID();

loadPumpID();

}

private void loadWellID()

{

try

{

cboWellID.Items.Clear();

cn.Open();

cmd.CommandText = "select Well\_ID from tblWellReg";

dr = cmd.ExecuteReader();

while (dr.Read())

cboWellID.Items.Add((dr["Well\_ID"]).ToString());

dr.Close();

cn.Close();

}

catch (Exception Er)

{

dr.Close();

cn.Close();

MessageBox.Show(Er.Message, "Failure Prediction App", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void loadPumpID()

{

try

{

cboPumpID.Items.Clear();

cn.Open();

cmd.CommandText = "select Pump\_ID from tblPumpReg";

dr = cmd.ExecuteReader();

while (dr.Read())

cboPumpID.Items.Add((dr["Pump\_ID"]).ToString());

dr.Close();

cn.Close();

}

catch (Exception Er)

{

dr.Close();

cn.Close();

MessageBox.Show(Er.Message, "Failure Prediction App", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (isAllEntryFilled())

{

string WellID = cboWellID.Text;

string PumpID = cboPumpID.Text;

string MeasuredDate = dtpMeasuredDate.Text;

string WellStatus = cboWellStatus.Text;

double CardArea = Convert.ToDouble(txtCardArea.Text);

double PeakSurfaceLoad = Convert.ToDouble(txtPeakSurfaceLoad.Text);

double MinimumSurfaceLoad = Convert.ToDouble(txtMinimumSurfaceLoad.Text);

double StrokePerMinute = Convert.ToDouble(txtStrokePerMinute.Text);

double SurfaceStrokeLength = Convert.ToDouble(txtSurfaceStrokeLength.Text);

double FlowLinePressure = Convert.ToDouble(txtFlowLinePressure.Text);

double PumpFillage = Convert.ToDouble(txtPumpFillage.Text);

double YesterdayCycles = Convert.ToDouble(txtYesterdayCycles.Text);

double DailyRuntime = Convert.ToDouble(txtDailyRuntime.Text);

double GearboxTorque = Convert.ToDouble(txtGearboxTorque.Text);

double PolishedRodHorsePower = Convert.ToDouble(txtPolishedRodHorsePower.Text);

double NetDownholePumpEfficiency = Convert.ToDouble(txtNetDownholePumpEfficiency.Text);

double LastApprovedOil = Convert.ToDouble(txtLastApprovedOil.Text);

double LastApprovedWater = Convert.ToDouble(txtLastApprovedWater.Text);

double FluidLevel = Convert.ToDouble(txtFluidLevel.Text);

double GrossFluidRate = Convert.ToDouble(txtGrossFluidRate.Text);

{

cn.Open();

cmd.CommandText = "insert into tblDailyMeasure values('" + WellID + "','" + PumpID + "','" + MeasuredDate + "','" + WellStatus + "'," + CardArea + "," + PeakSurfaceLoad + "," + MinimumSurfaceLoad + "," + StrokePerMinute + "," + SurfaceStrokeLength + ","

+ FlowLinePressure + "," + PumpFillage + "," + YesterdayCycles + "," + DailyRuntime + "," + GearboxTorque + "," + PolishedRodHorsePower + "," + NetDownholePumpEfficiency + "," + LastApprovedOil + "," + LastApprovedWater + "," + FluidLevel + "," + GrossFluidRate + ")";

cmd.ExecuteNonQuery();

MessageBox.Show("Daily Measure Multivariate Time Series Data Successfully Saved!!!", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Information);

cn.Close();

btnRefresh();

}

}

}

catch (Exception er)

{

cn.Close();

MessageBox.Show(er.Message, "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private bool isAllEntryFilled()

{

bool isEntryFilled = false;

if ((cboWellID.Text == ""))

{

MessageBox.Show("Please type the Well ID", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboWellID.Focus();

return isEntryFilled;

}

else if ((cboPumpID.Text == ""))

{

MessageBox.Show("Please type the Pump ID", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboPumpID.Focus();

return isEntryFilled;

}

else if ((cboWellStatus.Text == ""))

{

MessageBox.Show("Please type the Well Status", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

cboWellStatus.Focus();

return isEntryFilled;

}

else if ((txtCardArea.Text == ""))

{

MessageBox.Show("Please set the Card Area", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtCardArea.Focus();

return isEntryFilled;

}

else if ((txtPeakSurfaceLoad.Text == ""))

{

MessageBox.Show("Please set the Peak Surface Load", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtPeakSurfaceLoad.Focus();

return isEntryFilled;

}

else if ((txtMinimumSurfaceLoad.Text == ""))

{

MessageBox.Show("Please set the Minimum Surface Load", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtMinimumSurfaceLoad.Focus();

return isEntryFilled;

}

else if ((txtStrokePerMinute.Text == ""))

{

MessageBox.Show("Please type the Stroke Per Minute ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtStrokePerMinute.Focus();

return isEntryFilled;

}

else if ((txtSurfaceStrokeLength.Text == ""))

{

MessageBox.Show("Please type the Surface Stroke Length ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtSurfaceStrokeLength.Focus();

return isEntryFilled;

}

else if ((txtYesterdayCycles.Text == ""))

{

MessageBox.Show("Please type the Yesterday Cycles ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtYesterdayCycles.Focus();

return isEntryFilled;

}

else if ((txtDailyRuntime.Text == ""))

{

MessageBox.Show("Please type the Daily Runtime ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtDailyRuntime.Focus();

return isEntryFilled;

}

else if ((txtGrossFluidRate.Text == ""))

{

MessageBox.Show("Please type the Gross Fluid Rate ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtGrossFluidRate.Focus();

return isEntryFilled;

}

else

return true;

}

private void btnRefresh()

{

cboWellID.ResetText();

cboPumpID.ResetText();

dtpMeasuredDate.ResetText();

cboWellStatus.ResetText();

txtCardArea.ResetText();

txtPeakSurfaceLoad.ResetText();

txtMinimumSurfaceLoad.ResetText();

txtStrokePerMinute.ResetText();

txtSurfaceStrokeLength.ResetText();

txtFlowLinePressure.ResetText();

txtPumpFillage.ResetText();

txtYesterdayCycles.ResetText();

txtDailyRuntime.ResetText();

txtGearboxTorque.ResetText();

txtPolishedRodHorsePower.ResetText();

txtNetDownholePumpEfficiency.ResetText();

txtLastApprovedOil.ResetText();

txtLastApprovedWater.ResetText();

txtFluidLevel.ResetText();

txtGrossFluidRate.ResetText();

}

private void txtCardArea\_KeyDown(object sender, KeyEventArgs e)

{

TextBox txt = new TextBox();

txt = (TextBox)sender;

nonNumberEntered = false;

if (e.KeyCode < Keys.D0 || e.KeyCode > Keys.D9)

{

if (e.KeyCode < Keys.NumPad0 || e.KeyCode > Keys.NumPad9)

{

if ((e.KeyCode != Keys.Back) && (e.KeyCode != Keys.OemPeriod))

nonNumberEntered = true;

}

}

if (Control.ModifierKeys == Keys.Shift)

{

nonNumberEntered = true;

}

}

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

{

TextBox txt = new TextBox();

txt = (TextBox)sender;

if (nonNumberEntered == true) e.Handled = true; //capture keytrapping i.e. keypress = 0 (null)

}

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

{

this.Hide();

}

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

{

if ((txtCardArea.Text == ""))

{

MessageBox.Show("Please set the Card Area", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtCardArea.Focus();

}

else if ((txtPeakSurfaceLoad.Text == ""))

{

MessageBox.Show("Please set the Peak Surface Load", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtPeakSurfaceLoad.Focus();

}

else if ((txtMinimumSurfaceLoad.Text == ""))

{

MessageBox.Show("Please set the Minimum Surface Load", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtMinimumSurfaceLoad.Focus();

}

else if ((txtYesterdayCycles.Text == ""))

{

MessageBox.Show("Please type the Yesterday Cycles ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtYesterdayCycles.Focus();

}

else if ((txtDailyRuntime.Text == ""))

{

MessageBox.Show("Please type the Daily Runtime ", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

txtDailyRuntime.Focus();

}

else

{

double CardArea = Convert.ToDouble(txtCardArea.Text);

double PeakSurfaceLoad = Convert.ToDouble(txtPeakSurfaceLoad.Text);

double MinimumSurfaceLoad = Convert.ToDouble(txtMinimumSurfaceLoad.Text);

double YesterdayCycles = Convert.ToDouble(txtYesterdayCycles.Text);

double DailyRuntime = Convert.ToDouble(txtDailyRuntime.Text);

//check for well status at Normal level

if ((CardArea >= 6000 && CardArea <= 15000) && (PeakSurfaceLoad <= 20000 && MinimumSurfaceLoad >= 3500) && (YesterdayCycles >= 10 && YesterdayCycles <= 25) && (DailyRuntime >= 6 && DailyRuntime <= 12))

{

MessageBox.Show("Failure Prediction Result:\n\r There is no failure detected.\n\r Well Status: Normal", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Information);

cboWellStatus.Text = "Normal";

}

else

{

MessageBox.Show("Failure Prediction Result:\n\r There is failure detected.\n\r Well Status: Fail", "Failure Prediction App.", MessageBoxButtons.OK, MessageBoxIcon.Information);

cboWellStatus.Text = "Fail";

}

}

}

}

}