# **Dashboard**

using Google.Apis.Auth.OAuth2;

using Google.Apis.Download;

using Google.Apis.Drive.v3;

using Google.Apis.Drive.v3.Data;

using Google.Apis.Services;

using Google.Apis.Upload;

using Google.Apis.Util.Store;

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 System.Data.OleDb;

using System.IO;

namespace Insert

{

public partial class Dashboard : Form

{

OleDbConnection cnon = new OleDbConnection();

private static string json\_secret\_file = @".\client\_secret.json";

private static string application\_name = @".CRM";

OleDbCommand com3;

string str3;

public Dashboard()

{

InitializeComponent();

cnon.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=|DataDirectory|\Contacts.accdb";

}

private void downloadFile(string fileName, string fileID, string mimeType)

{

string path = Path.GetDirectoryName(Application.ExecutablePath);

Task.Run(() => { CloudDownload.downloadFromDrive(fileName, fileID, path, mimeType, this); });

}

private void Form1\_Load(object sender, EventArgs e)

{

CloudDownload driveDownload = new CloudDownload(json\_secret\_file, application\_name);

string fileName, fileId, mimeType;

fileName = "Contacts.accdb";

mimeType = "application/msaccess";

fileId = "1rRvc7aKnzKVvvTBwtOnT1-1GO2kz6iK9";

downloadFile(fileName, fileId, mimeType);

str3 = "SELECT Item,Sold from ItemSales";

com3 = new OleDbCommand(str3, cnon);

OleDbDataAdapter da = new OleDbDataAdapter(com3);

DataTable dt = new DataTable();

da.Fill(dt);

cnon.Close();

string[] N = new string[dt.Rows.Count];

int[] M = new int[dt.Rows.Count];

for (int i = 0; i < dt.Rows.Count; i++)

{

N[i] = dt.Rows[i][0].ToString();

M[i] = Convert.ToInt32(dt.Rows[i][1]);

}

chart1.Series[0].Points.DataBindXY(N, M);

}

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

{

this.Hide();

var Invent = new InventManagement();

Invent.Closed += (s, args) => this.Close();

Invent.Show();

}

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

{

this.Hide();

var Task = new TasksForm();

Task.Closed += (s, args) => this.Close();

Task.Show();

}

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

{

this.Hide();

var contact = new ContactsForm();

contact.Closed += (s, args) => this.Close();

contact.Show();

}

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

{

this.Hide();

var cal = new Calendar();

cal.Closed += (s, args) => this.Close();

cal.Show();

}

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

{

this.Hide();

var emailf = new EmailForm();

emailf.Closed += (s, args) => this.Close();

emailf.Show();

}

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

{

if (comboBox1.Text == "Item Sales")

{

str3 = "SELECT Item,Sold from ItemSales";

com3 = new OleDbCommand(str3, cnon);

OleDbDataAdapter da = new OleDbDataAdapter(com3);

DataTable dt = new DataTable();

da.Fill(dt);

cnon.Close();

string[] N = new string[dt.Rows.Count];

int[] M = new int[dt.Rows.Count];

for (int i = 0; i < dt.Rows.Count; i++)

{

N[i] = dt.Rows[i][0].ToString();

M[i] = Convert.ToInt32(dt.Rows[i][1]);

}

chart1.Series[0].Points.DataBindXY(N, M);

}

else if (comboBox1.Text == "Monthly Sales")

{

str3 = "SELECT Months,TotalSales from Sales";

com3 = new OleDbCommand(str3, cnon);

OleDbDataAdapter da = new OleDbDataAdapter(com3);

DataTable dt = new DataTable();

da.Fill(dt);

cnon.Close();

string[] N = new string[dt.Rows.Count];

int[] M = new int[dt.Rows.Count];

for (int i = 0; i < dt.Rows.Count; i++)

{

N[i] = dt.Rows[i][0].ToString();

M[i] = Convert.ToInt32(dt.Rows[i][1]);

}

chart1.Series[0].Points.DataBindXY(N, M);

chart1.Series[0].LegendText = "Total Sales";

}

}

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

{

this.Hide();

var Cust = new CustomersForm();

Cust.Closed += (s, args) => this.Close();

Cust.Show();

}

}

}

# **Contacts Form**

using Google.Apis.Auth.OAuth2;

using Google.Apis.Drive.v3;

using Google.Apis.Drive.v3.Data;

using Google.Apis.Services;

using Google.Apis.Util.Store;

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 System.Data.OleDb;

using System.IO;

namespace Insert

{

public partial class ContactsForm : Form

{

//

//Establish the database connection

//

OleDbConnection cnon = new OleDbConnection();

//

//

//

private static string json\_secret\_file = @".\client\_secret.json";

private static string application\_name = @".GoogleDriveUploader";

private string GetFileType(string file)

{

string extension = Path.GetExtension(file);

System.Diagnostics.Debug.WriteLine("extension: " + extension);

string mime;

switch (extension.ToLower())

{

case ".jpg":

mime = "image/jpeg";

break;

case ".jpeg":

mime = "image/jpeg";

break;

case ".png":

mime = "image/png";

break;

case ".accdb":

mime = "application/msaccess";

break;

case ".pptx":

mime = "application/vnd.google-apps.presentation";

break;

case ".xlsx":

mime = "application/vnd.google-apps.spreadsheet";

break;

case ".doc":

mime = "application/vnd.google-apps.document";

break;

case ".docx":

mime = "application/vnd.google-apps.document";

break;

case ".pdf":

mime = "application/pdf";

break;

default:

mime = "text/plain";

break;

}

return mime;

}

public ContactsForm()

{

InitializeComponent();

//

//Pathway to the database

//

cnon.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=|DataDirectory|\Contacts.accdb";

//

//

//

}

//

//Insert data into the database

//

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

{

if (Companytxt.Text == "")

{

MessageBox.Show("Please fill in the Company box");

}

else if(Emailtxt.Text == "")

{

MessageBox.Show("Please fill in the Email box");

}

else if (Citytxt.Text == "")

{

MessageBox.Show("Please fill in the City box");

}

else if (Countrytxt.Text == "")

{

MessageBox.Show("Please fill in the Country box");

}

else

{

OleDbCommand command = new OleDbCommand("INSERT INTO Contacts (Company,LastName , FirstName, Email, JobTitle, BusinessPhone, HomePhone, MobilePhone, FaxNumber, Address, City, State, Postcode, Country, Webpage, Notes) VALUES(?, ?, ? ,?, ?, ?, ? ,?, ?, ?, ? ,?, ?, ?, ? ,?)", cnon);

command.Parameters.AddWithValue("@Company", Companytxt.Text);

command.Parameters.AddWithValue("@LastName", LastNametxt.Text);

command.Parameters.AddWithValue("@FirstName", FirstNametxt.Text);

command.Parameters.AddWithValue("@Email", Emailtxt.Text);

command.Parameters.AddWithValue("@JobTitle", Jobtxt.Text);

command.Parameters.AddWithValue("@BusinessPhone", BPhonetxt.Text);

command.Parameters.AddWithValue("@HomePhone", Hphonetxt.Text);

command.Parameters.AddWithValue("@MobilePhone", Mphonetxt.Text);

command.Parameters.AddWithValue("@FaxNumber", Faxnotxt.Text);

command.Parameters.AddWithValue("@Address", Addresstxt.Text);

command.Parameters.AddWithValue("@City", Citytxt.Text);

command.Parameters.AddWithValue("@State", Statetxt.Text);

command.Parameters.AddWithValue("@Postcode", Postcodetxt.Text);

command.Parameters.AddWithValue("@Country", Countrytxt.Text);

command.Parameters.AddWithValue("@Webpage", webpagetxt.Text);

command.Parameters.AddWithValue("@Notes", Notestxt.Text);

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("Save Successful");

cnon.Close();

SaveBtn.Enabled = false;

EditBtn.Enabled = true;

DeleteBtn.Enabled = true;

Companytxt.Text = "";

LastNametxt.Text = "";

FirstNametxt.Text = "";

Emailtxt.Text = "";

Jobtxt.Text = "";

BPhonetxt.Text = "";

Hphonetxt.Text = "";

Mphonetxt.Text = "";

Faxnotxt.Text = "";

Addresstxt.Text = "";

Citytxt.Text = "";

Statetxt.Text = "";

Postcodetxt.Text = "";

Countrytxt.Text = "";

webpagetxt.Text = "";

Notestxt.Text = "";

//

//

//

}

//

//Adds the new record into the datagridview by loading the database again.

//

string query = "SELECT \* From Contacts";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

ContactGridList.DataSource = ds.Tables[0];

}

//

//

//

}

//

//Load the database into the datagridview

//

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

{

string query = "SELECT \* From Contacts";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

ContactGridList.DataSource = ds.Tables[0];

}

}

//

//

//

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

{

AddBtn.Enabled = false;

DeleteBtn.Enabled = false;

UpdateBtn.Enabled = true;

FirstNametxt.ReadOnly = false;

LastNametxt.ReadOnly = false;

Companytxt.ReadOnly = false;

Emailtxt.ReadOnly = false;

Jobtxt.ReadOnly = false;

BPhonetxt.ReadOnly = false;

Hphonetxt.ReadOnly = false;

Mphonetxt.ReadOnly = false;

Faxnotxt.ReadOnly = false;

Addresstxt.ReadOnly = false;

Citytxt.ReadOnly = false;

Statetxt.ReadOnly = false;

Postcodetxt.ReadOnly = false;

Countrytxt.ReadOnly = false;

webpagetxt.ReadOnly = false;

Notestxt.ReadOnly = false;

}

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

{

if (IDtxt.Text == "")

{

MessageBox.Show("Select a record");

}

else

{

string query = "DELETE from Contacts WHERE ID=" + IDtxt.Text;

using (OleDbCommand command = new OleDbCommand(query, cnon))

{

command.CommandType = CommandType.Text;

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("Delete Successful");

cnon.Close();

}

IDtxt.Text = "";

Companytxt.Text = "";

LastNametxt.Text = "";

FirstNametxt.Text = "";

Emailtxt.Text = "";

Jobtxt.Text = "";

BPhonetxt.Text = "";

Hphonetxt.Text = "";

Mphonetxt.Text = "";

Faxnotxt.Text = "";

Addresstxt.Text = "";

Citytxt.Text = "";

Statetxt.Text = "";

Postcodetxt.Text = "";

Countrytxt.Text = "";

webpagetxt.Text = "";

Notestxt.Text = "";

}

string query2 = "SELECT \* From Contacts";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query2, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

ContactGridList.DataSource = ds.Tables[0];

}

}

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

{

SaveBtn.Enabled = true;

UpdateBtn.Enabled = false;

EditBtn.Enabled = false;

DeleteBtn.Enabled = false;

FirstNametxt.ReadOnly = false;

LastNametxt.ReadOnly = false;

Companytxt.ReadOnly = false;

Emailtxt.ReadOnly = false;

Jobtxt.ReadOnly = false;

BPhonetxt.ReadOnly = false;

Hphonetxt.ReadOnly = false;

Mphonetxt.ReadOnly = false;

Faxnotxt.ReadOnly = false;

Addresstxt.ReadOnly = false;

Citytxt.ReadOnly = false;

Statetxt.ReadOnly = false;

Postcodetxt.ReadOnly = false;

Countrytxt.ReadOnly = false;

webpagetxt.ReadOnly = false;

Notestxt.ReadOnly = false;

IDtxt.Text = "";

Companytxt.Text = "";

LastNametxt.Text = "";

FirstNametxt.Text = "";

Emailtxt.Text = "";

Jobtxt.Text = "";

BPhonetxt.Text = "";

Hphonetxt.Text = "";

Mphonetxt.Text = "";

Faxnotxt.Text = "";

Addresstxt.Text = "";

Citytxt.Text = "";

Statetxt.Text = "";

Postcodetxt.Text = "";

Countrytxt.Text = "";

webpagetxt.Text = "";

Notestxt.Text = "";

}

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

{

//it checks if the row index of the cell is greater than or equal to zero

if (e.RowIndex >= 0)

{

//gets a collection that contains all the rows

DataGridViewRow row = this.ContactGridList.Rows[e.RowIndex];

//populate the textbox from specific value of the coordinates of column and row.

IDtxt.Text = row.Cells[0].Value.ToString();

Companytxt.Text = row.Cells[1].Value.ToString();

LastNametxt.Text = row.Cells[2].Value.ToString();

FirstNametxt.Text = row.Cells[3].Value.ToString();

Emailtxt.Text = row.Cells[4].Value.ToString();

Jobtxt.Text = row.Cells[5].Value.ToString();

BPhonetxt.Text = row.Cells[6].Value.ToString();

Hphonetxt.Text = row.Cells[7].Value.ToString();

Mphonetxt.Text = row.Cells[8].Value.ToString();

Faxnotxt.Text = row.Cells[9].Value.ToString();

Addresstxt.Text = row.Cells[10].Value.ToString();

Citytxt.Text = row.Cells[11].Value.ToString();

Statetxt.Text = row.Cells[12].Value.ToString();

Postcodetxt.Text = row.Cells[13].Value.ToString();

Countrytxt.Text = row.Cells[14].Value.ToString();

webpagetxt.Text = row.Cells[15].Value.ToString();

Notestxt.Text = row.Cells[16].Value.ToString();

}

}

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

{

if (Companytxt.Text == "")

{

MessageBox.Show("Please fill in the Company box");

}

else if (Emailtxt.Text == "")

{

MessageBox.Show("Please fill in the Email box");

}

else if (Citytxt.Text == "")

{

MessageBox.Show("Please fill in the City box");

}

else if (Countrytxt.Text == "")

{

MessageBox.Show("Please fill in the Country box");

}

else

{

String SQLstring = "UPDATE Contacts Set Company = ?, LastName = ?, FirstName = ?, Email = ?, JobTitle = ?, BusinessPhone = ?, HomePhone = ?, MobilePhone = ?, FaxNumber = ?, Address = ?, City = ?, State = ?, Postcode = ?, Country = ?, Webpage = ?, Notes =? WHERE ID = " + IDtxt.Text + "";

using (OleDbCommand command = new OleDbCommand(SQLstring, cnon))

{

command.CommandType = CommandType.Text;

command.Parameters.AddWithValue("@Company", Companytxt.Text);

command.Parameters.AddWithValue("@LastName", LastNametxt.Text);

command.Parameters.AddWithValue("@FirstName", FirstNametxt.Text);

command.Parameters.AddWithValue("@Email", Emailtxt.Text);

command.Parameters.AddWithValue("@JobTitle", Jobtxt.Text);

command.Parameters.AddWithValue("@BusinessPhone", BPhonetxt.Text);

command.Parameters.AddWithValue("@HomePhone", Hphonetxt.Text);

command.Parameters.AddWithValue("@MobilePhone", Mphonetxt.Text);

command.Parameters.AddWithValue("@FaxNumber", Faxnotxt.Text);

command.Parameters.AddWithValue("@Address", Addresstxt.Text);

command.Parameters.AddWithValue("@City", Citytxt.Text);

command.Parameters.AddWithValue("@State", Statetxt.Text);

command.Parameters.AddWithValue("@Postcode", Postcodetxt.Text);

command.Parameters.AddWithValue("@Country", Countrytxt.Text);

command.Parameters.AddWithValue("@Webpage", webpagetxt.Text);

command.Parameters.AddWithValue("@Notes", Notestxt.Text);

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("Update Successful");

cnon.Close();

}

}

UpdateBtn.Enabled = false;

SaveBtn.Enabled = false;

AddBtn.Enabled = true;

DeleteBtn.Enabled = true;

FirstNametxt.ReadOnly = true;

LastNametxt.ReadOnly = true;

Companytxt.ReadOnly = true;

Emailtxt.ReadOnly = true;

Jobtxt.ReadOnly = true;

BPhonetxt.ReadOnly = true;

Hphonetxt.ReadOnly = true;

Mphonetxt.ReadOnly = true;

Faxnotxt.ReadOnly = true;

Addresstxt.ReadOnly = true;

Citytxt.ReadOnly = true;

Statetxt.ReadOnly = true;

Postcodetxt.ReadOnly = true;

Countrytxt.ReadOnly = true;

webpagetxt.ReadOnly = true;

Notestxt.ReadOnly = true;

IDtxt.Text = "";

Companytxt.Text = "";

LastNametxt.Text = "";

FirstNametxt.Text = "";

Emailtxt.Text = "";

Jobtxt.Text = "";

BPhonetxt.Text = "";

Hphonetxt.Text = "";

Mphonetxt.Text = "";

Faxnotxt.Text = "";

Addresstxt.Text = "";

Citytxt.Text = "";

Statetxt.Text = "";

Postcodetxt.Text = "";

Countrytxt.Text = "";

webpagetxt.Text = "";

Notestxt.Text = "";

string query = "SELECT \* From Contacts";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

ContactGridList.DataSource = ds.Tables[0];

}

}

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

{

string pathfile = Path.GetDirectoryName(Application.ExecutablePath) + @"\Contacts.accdb";

string filename = "Contacts.accdb";

CloudUpload driveUploader = new CloudUpload(json\_secret\_file, application\_name);

System.Diagnostics.Debug.WriteLine(pathfile);

try

{

byte[] byteArray = System.IO.File.ReadAllBytes(pathfile);

string filePath = pathfile;

string fileName = filename;

string fileId = "1rRvc7aKnzKVvvTBwtOnT1-1GO2kz6iK9";

string description = "test";

string fileType = GetFileType(filePath);

driveUploader.UploadFile(ref byteArray, fileName, fileId, description, fileType);

}

catch (Exception exc)

{

System.Diagnostics.Debug.WriteLine(exc.Message);

}

}

# Sync

public void sync()

{

string pathfile = Path.GetDirectoryName(Application.ExecutablePath) + @"\Contacts.accdb";

string filename = "Contacts.accdb";

CloudUpload driveUploader = new CloudUpload(json\_secret\_file, application\_name);

System.Diagnostics.Debug.WriteLine(pathfile);

try

{

byte[] byteArray = System.IO.File.ReadAllBytes(pathfile);

string filePath = pathfile;

string fileName = filename;

string fileId = "1rRvc7aKnzKVvvTBwtOnT1-1GO2kz6iK9";

string description = "test";

string fileType = GetFileType(filePath);

driveUploader.UploadFile(ref byteArray, fileName, fileId, description, fileType);

}

catch (Exception exc)

{

System.Diagnostics.Debug.WriteLine(exc.Message);

}

}

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

{

SaveBtn.Enabled = false;

AddBtn.Enabled = true;

UpdateBtn.Enabled = true;

EditBtn.Enabled = true;

FirstNametxt.ReadOnly = true;

LastNametxt.ReadOnly = true;

Companytxt.ReadOnly = true;

Emailtxt.ReadOnly = true;

Jobtxt.ReadOnly = true;

BPhonetxt.ReadOnly = true;

Hphonetxt.ReadOnly = true;

Mphonetxt.ReadOnly = true;

Faxnotxt.ReadOnly = true;

Addresstxt.ReadOnly = true;

Citytxt.ReadOnly = true;

Statetxt.ReadOnly = true;

Postcodetxt.ReadOnly = true;

Countrytxt.ReadOnly = true;

webpagetxt.ReadOnly = true;

Notestxt.ReadOnly = true;

IDtxt.Text = "";

Companytxt.Text = "";

LastNametxt.Text = "";

FirstNametxt.Text = "";

Emailtxt.Text = "";

Jobtxt.Text = "";

BPhonetxt.Text = "";

Hphonetxt.Text = "";

Mphonetxt.Text = "";

Faxnotxt.Text = "";

Addresstxt.Text = "";

Citytxt.Text = "";

Statetxt.Text = "";

Postcodetxt.Text = "";

Countrytxt.Text = "";

webpagetxt.Text = "";

Notestxt.Text = "";

}

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

{

sync();

this.Hide();

var dash = new Dashboard();

dash.Closed += (s, args) => this.Close();

dash.Show();

}

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

{

sync();

this.Hide();

var cal = new Calendar();

cal.Closed += (s, args) => this.Close();

cal.Show();

}

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

{

sync();

this.Hide();

var task = new TasksForm();

task.Closed += (s, args) => this.Close();

task.Show();

}

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

{

sync();

this.Hide();

var invent = new InventManagement();

invent.Closed += (s, args) => this.Close();

invent.Show();

}

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

{

cnon.Open();

string query = "Select \* from Contacts where FirstName like '" + SearchTxt.Text + "%'";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

ContactGridList.DataSource = ds.Tables[0];

cnon.Close();

}

}

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

{

this.Hide();

var Cust = new CustomersForm();

Cust.Closed += (s, args) => this.Close();

Cust.Show();

}

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

{

this.Hide();

var emailf = new EmailForm();

emailf.Closed += (s, args) => this.Close();

emailf.Show();

}

}

}

# **Calendar**

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 System.Data.OleDb;

using System.IO;

namespace Insert

{

public partial class Calendar : Form

{

//

//Establish the database connection

//

OleDbConnection cnon = new OleDbConnection();

//

//

//

private static string json\_secret\_file = @".\client\_secret.json";

private static string application\_name = @".GoogleDriveUploader";

private string GetFileType(string file)

{

string extension = Path.GetExtension(file);

System.Diagnostics.Debug.WriteLine("extension: " + extension);

string mime;

switch (extension.ToLower())

{

case ".jpg":

mime = "image/jpeg";

break;

case ".jpeg":

mime = "image/jpeg";

break;

case ".png":

mime = "image/png";

break;

case ".accdb":

mime = "application/msaccess";

break;

case ".pptx":

mime = "application/vnd.google-apps.presentation";

break;

case ".xlsx":

mime = "application/vnd.google-apps.spreadsheet";

break;

case ".doc":

mime = "application/vnd.google-apps.document";

break;

case ".docx":

mime = "application/vnd.google-apps.document";

break;

case ".pdf":

mime = "application/pdf";

break;

default:

mime = "text/plain";

break;

}

return mime;

}

public Calendar()

{

InitializeComponent();

//Pathway to the database

//

cnon.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=|DataDirectory|\Contacts.accdb";

//

//

// To display single selected of date

monthCalendar1.MaxSelectionCount = 1;

}

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

{

string query = "SELECT \* From Events";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

SelectedDateDGV.DataSource = ds.Tables[0];

}

}

private void monthCalendar1\_DateChanged(object sender, DateRangeEventArgs e)

{

// To display single selected of date

monthCalendar1.MaxSelectionCount = 1;

// To display single date use MonthCalendar1.SelectionRange.Start/ MonthCalendarSelectionRange.End

SelectedDateTB.Text = monthCalendar1.SelectionRange.End.ToShortDateString();

//string query = "SELECT \* From Events WHERE EventDate = " + SelectedDateTB.Text;

using (OleDbDataAdapter adapter = new OleDbDataAdapter("SELECT \* From Events WHERE EventDate =#" + SelectedDateTB.Text + "#" , cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

SelectedDateDGV.DataSource = ds.Tables[0];

}

}

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

{

if (EventNameTB.Text == ""|Durationtxt.Text=="Input Integer"|DurationCB.Text=="")

{

MessageBox.Show("Please fill in all boxes");

}

else

{

OleDbCommand command = new OleDbCommand("INSERT INTO Events (EventName, EventDesc, EventDate, Duration) VALUES(?, ? ,?, ?)", cnon);

command.Parameters.AddWithValue("@EventName", EventNameTB.Text);

command.Parameters.AddWithValue("@EventDesc", EventDescTB.Text);

command.Parameters.AddWithValue("@EventDate", StartDateDTP.Value.ToString());

command.Parameters.AddWithValue("@Duration", Durationtxt.Text + " " + DurationCB.Text);

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("WELL DONE");

cnon.Close();

SaveBtn.Enabled = false;

AddBtn.Enabled = true;

UpdateBtn.Enabled = false;

EditBtn.Enabled = true;

DeleteBtn.Enabled = true;

EventNameTB.Enabled = false;

EventDescTB.Enabled = false;

StartDateDTP.Enabled = false;

Durationtxt.Enabled = false;

DurationCB.Enabled = false;

IDTB.Text = "";

EventNameTB.Text = "";

EventDescTB.Text = "";

Durationtxt.Text = "";

//

//Adds the new record into the datagridview by loading the database again.

//

string query = "SELECT \* From Events";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

SelectedDateDGV.DataSource = ds.Tables[0];

}

//

//

//

}

}

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

{

}

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

{

if (e.RowIndex >= 0)

{

//gets a collection that contains all the rows

DataGridViewRow row = this.SelectedDateDGV.Rows[e.RowIndex];

//populate the textbox from specific value of the coordinates of column and row.

IDTB.Text = row.Cells[0].Value.ToString();

EventNameTB.Text = row.Cells[1].Value.ToString();

EventDescTB.Text = row.Cells[2].Value.ToString();

StartDateDTP.Text = row.Cells[3].Value.ToString();

Durationtxt.Text = row.Cells[4].Value.ToString();

}

}

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

{

SaveBtn.Enabled = true;

EditBtn.Enabled = false;

DeleteBtn.Enabled = false;

EventNameTB.Enabled = true;

EventDescTB.Enabled = true;

StartDateDTP.Enabled = true;

Durationtxt.Enabled = true;

DurationCB.Enabled = true;

IDTB.Text = "";

EventNameTB.Text = "";

EventDescTB.Text = "";

}

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

{

}

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

{

if (IDTB.Text == "")

{

MessageBox.Show("Select a record)");

}

else

{

string query = "DELETE from Events WHERE EventID=" + IDTB.Text;

using (OleDbCommand command = new OleDbCommand(query, cnon))

{

command.CommandType = CommandType.Text;

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("Delete Successful");

cnon.Close();

}

IDTB.Text = "";

EventNameTB.Text = "";

EventDescTB.Text = "";

Durationtxt.Text = "";

string query2 = "SELECT \* From Events";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query2, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

SelectedDateDGV.DataSource = ds.Tables[0];

}

}

}

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

{

if (Durationtxt.Text == "Input Integer")

{

Durationtxt.Text = "";

}

else

{

Durationtxt.Text = "Input Integer";

}

}

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

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&

(e.KeyChar != '.'))

{

e.Handled = true;

}

}

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

{

AddBtn.Enabled = false;

DeleteBtn.Enabled = false;

UpdateBtn.Enabled = true;

EventNameTB.Enabled = true;

EventDescTB.Enabled = true;

StartDateDTP.Enabled = true;

Durationtxt.Enabled = true;

DurationCB.Enabled = true;

}

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

{

if (EventNameTB.Text == "" | EventDescTB.Text == "" | Durationtxt.Text == "Input Integer" | DurationCB.Text == "")

{

MessageBox.Show("Please fill in all boxes");

}

else

{

string sqlquery = "UPDATE Events SET EventName = ?, EventDesc = ?, EventDate = ?, Duration = ? WHERE EventID = " + IDTB.Text + "";

using (OleDbCommand command = new OleDbCommand(sqlquery, cnon))

{

command.Parameters.AddWithValue("@EventName", EventNameTB.Text);

command.Parameters.AddWithValue("@EventDesc", EventDescTB.Text);

command.Parameters.AddWithValue("@EventDate", StartDateDTP.Value.ToString());

command.Parameters.AddWithValue("@Duration", Durationtxt.Text + " " + DurationCB.Text);

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("Update Successful");

cnon.Close();

}

IDTB.Text = "";

EventNameTB.Text = "";

EventDescTB.Text = "";

Durationtxt.Text = "";

AddBtn.Enabled = true;

DeleteBtn.Enabled = false;

UpdateBtn.Enabled = false;

//

//Adds the new record into the datagridview by loading the database again.

//

string query = "SELECT \* From Events";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

SelectedDateDGV.DataSource = ds.Tables[0];

}

//

//

//

}

}

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

{

SaveBtn.Enabled = false;

AddBtn.Enabled = true;

UpdateBtn.Enabled = true;

EditBtn.Enabled = true;

DeleteBtn.Enabled = true;

EventNameTB.Enabled = false;

EventDescTB.Enabled = false;

StartDateDTP.Enabled = false;

Durationtxt.Enabled = false;

DurationCB.Enabled = false;

IDTB.Text = "";

EventNameTB.Text = "";

EventDescTB.Text = "";

Durationtxt.Text = "";

}

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

{

this.Hide();

var dashb = new Dashboard();

dashb.Closed += (s, args) => this.Close();

dashb.Show();

}

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

{

this.Hide();

var contact = new ContactsForm();

contact.Closed += (s, args) => this.Close();

contact.Show();

}

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

{

this.Hide();

var Task = new TasksForm();

Task.Closed += (s, args) => this.Close();

Task.Show();

}

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

{

this.Hide();

var Invent = new InventManagement();

Invent.Closed += (s, args) => this.Close();

Invent.Show();

}

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

{

this.Hide();

var emailf = new EmailForm();

emailf.Closed += (s, args) => this.Close();

emailf.Show();

}

public void sync()

{

string pathfile = Path.GetDirectoryName(Application.ExecutablePath) + @"\Contacts.accdb";

string filename = "Contacts.accdb";

CloudUpload driveUploader = new CloudUpload(json\_secret\_file, application\_name);

System.Diagnostics.Debug.WriteLine(pathfile);

try

{

byte[] byteArray = System.IO.File.ReadAllBytes(pathfile);

string filePath = pathfile;

string fileName = filename;

string fileId = "1rRvc7aKnzKVvvTBwtOnT1-1GO2kz6iK9";

string description = "test";

string fileType = GetFileType(filePath);

driveUploader.UploadFile(ref byteArray, fileName, fileId, description, fileType);

}

catch (Exception exc)

{

System.Diagnostics.Debug.WriteLine(exc.Message);

}

}

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

{

this.Hide();

var Cust = new CustomersForm();

Cust.Closed += (s, args) => this.Close();

Cust.Show();

}

}

}

# **Cloud Download**

//Used this tutorial to guide me and learn how to implement Google drive https://www.daimto.com/google-drive-api-c-upload/

using Google.Apis.Auth.OAuth2;

using Google.Apis.Download;

using Google.Apis.Drive.v3;

using Google.Apis.Drive.v3.Data;

using Google.Apis.Services;

using Google.Apis.Upload;

using Google.Apis.Util.Store;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Insert

{

class CloudDownload

{

public static UserCredential credential;

public static DriveService driveService;

public CloudDownload(string jsonSecretPath, string appName)

{

GetCredential(jsonSecretPath);

CreateDriveService(appName);

}

private void GetCredential(string clientSecretPath)

{

using (var filestream = new FileStream(clientSecretPath,

FileMode.Open, FileAccess.Read))

{

credential = GoogleWebAuthorizationBroker.AuthorizeAsync(

GoogleClientSecrets.Load(filestream).Secrets,

new[] { DriveService.Scope.Drive },

"user",

CancellationToken.None,

new FileDataStore("DriveCommandLineSample")).Result;

}

}

private void CreateDriveService(string applicationName)

{

driveService = new DriveService(new BaseClientService.Initializer()

{

HttpClientInitializer = credential,

ApplicationName = applicationName,

});

}

private static void convertMemoryStreamToFileStream(MemoryStream stream, string savePath)

{

FileStream fileStream;

using (fileStream = new System.IO.FileStream(savePath, FileMode.OpenOrCreate, FileAccess.Write))

{

try

{

// System.IO.File.Create(saveFile)

stream.WriteTo(fileStream);

fileStream.Close();

}

catch (Exception exc)

{

System.Diagnostics.Debug.WriteLine(exc.Message + " Convert Memory stream Error");

}

}

}

public static void downloadFromDrive(string filename, string fileId, string savePath, string mimeType, Dashboard parentForm)

{

Dashboard parent = parentForm;

try

{

if (Path.HasExtension(filename))

{

var request = driveService.Files.Get(fileId);

var stream = new System.IO.MemoryStream();

System.Diagnostics.Debug.WriteLine(fileId);

request.MediaDownloader.ProgressChanged +=

(IDownloadProgress progress) =>

{

switch (progress.Status)

{

case DownloadStatus.Downloading:

{

System.Diagnostics.Debug.WriteLine(progress.BytesDownloaded);

break;

}

case DownloadStatus.Completed:

{

System.Diagnostics.Debug.WriteLine("Download complete.");

break;

}

case DownloadStatus.Failed:

{

System.Diagnostics.Debug.WriteLine("Download failed.");

MessageBox.Show("File failed to download!!!", "Download Message", MessageBoxButtons.OK, MessageBoxIcon.Error);

break;

}

}

};

request.Download(stream);

convertMemoryStreamToFileStream(stream, savePath + @"\" + @filename);

stream.Dispose();

}

else

{

string extension = "", converter = "";

foreach (MimeTypeConvert obj in MimeConverter.mimeList())

{

if (mimeType == obj.mimeType)

{

extension = obj.extension;

converter = obj.converterType;

}

}

System.Diagnostics.Debug.WriteLine("{0} {1} {2}", fileId, extension, mimeType);

var request = driveService.Files.Export(fileId, converter);

var stream = new System.IO.MemoryStream();

request.MediaDownloader.ProgressChanged +=

(IDownloadProgress progress) =>

{

switch (progress.Status)

{

case DownloadStatus.Downloading:

{

Console.WriteLine(progress.BytesDownloaded);

break;

}

case DownloadStatus.Completed:

{

Console.WriteLine("Download complete.");

break;

}

case DownloadStatus.Failed:

{

Console.WriteLine("Download failed.");

MessageBox.Show("File failed to download!!!", "Download Message", MessageBoxButtons.OK, MessageBoxIcon.Error);

break;

}

}

};

request.Download(stream);

convertMemoryStreamToFileStream(stream, savePath + @"\" + @filename + extension);

stream.Dispose();

}

}

catch (Exception exc)

{

System.Diagnostics.Debug.WriteLine(exc.Message + " Download From Drive Error");

}

}

}

}

//Used this tutorial to guide me and learn how to implement Google drive https://www.daimto.com/google-drive-api-c-upload/

using Google.Apis.Auth.OAuth2;

using Google.Apis.Drive.v3;

using Google.Apis.Drive.v3.Data;

using Google.Apis.Services;

using Google.Apis.Util.Store;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Insert

{

class CloudUpload

{

private UserCredential credential;

private DriveService driveService;

public CloudUpload(string jsonSecretPath, string appName)

{

GetCredential(jsonSecretPath);

CreateDriveService(appName);

}

private void GetCredential(string clientSecretPath)

{

using (var filestream = new FileStream(clientSecretPath,

FileMode.Open, FileAccess.Read))

{

credential = GoogleWebAuthorizationBroker.AuthorizeAsync(

GoogleClientSecrets.Load(filestream).Secrets,

new[] { DriveService.Scope.Drive },

"user",

CancellationToken.None,

new FileDataStore("DriveCommandLineSample")).Result;

}

}

private void CreateDriveService(string applicationName)

{

driveService = new DriveService(new BaseClientService.Initializer()

{

HttpClientInitializer = credential,

ApplicationName = applicationName,

});

}

private void uploadDocumentToDrive(ref byte[] file, string title, string fileId, string description, string mimetype)

{

Google.Apis.Drive.v3.Data.File body = new Google.Apis.Drive.v3.Data.File();

body.Name = title;

body.Description = description;

body.MimeType = mimetype;

fileId = "1rRvc7aKnzKVvvTBwtOnT1-1GO2kz6iK9";

using (System.IO.MemoryStream stream = new System.IO.MemoryStream(file))

{

if (driveService != null)

{

FilesResource.UpdateMediaUpload request = driveService.Files.Update(body, fileId, stream, mimetype);

request.Upload();

Google.Apis.Drive.v3.Data.File uploadedFile = request.ResponseBody;

System.Diagnostics.Debug.WriteLine("Uploaded file: {0} with ID: {1}",

uploadedFile.Name,

uploadedFile.Id);

}

}

}

public bool UploadFile(ref byte[] file, string title, string fileId, string description, string mimetype)

{

bool uploaded = false;

if (credential != null)

{

uploadDocumentToDrive(ref file, title, fileId, description, mimetype);

uploaded = true;

}

return uploaded;

}

}

}

# **Customer Form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.OleDb;

namespace Insert

{

public partial class CustomersForm : Form

{

OleDbConnection cnon = new OleDbConnection();

public CustomersForm()

{

InitializeComponent();

cnon.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=|DataDirectory|\Contacts.accdb";

}

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

{

string query = "SELECT \* From Customer";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

CustomerDGV.DataSource = ds.Tables[0];

}

}

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

{

if (comboBox1.Text == "")

{

MessageBox.Show("Please select what column you want to filter in the combo box");

}

else

{

cnon.Open();

string query = "Select \* from Customer where " + comboBox1.Text + " like '" + SearchTxt.Text + "%'";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

CustomerDGV.DataSource = ds.Tables[0];

cnon.Close();

}

}

}

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

{

this.Hide();

var Invent = new InventManagement();

Invent.Closed += (s, args) => this.Close();

Invent.Show();

}

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

{

this.Hide();

var dashb = new Dashboard();

dashb.Closed += (s, args) => this.Close();

dashb.Show();

}

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

{

this.Hide();

var contact = new ContactsForm();

contact.Closed += (s, args) => this.Close();

contact.Show();

}

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

{

this.Hide();

var cal = new Calendar();

cal.Closed += (s, args) => this.Close();

cal.Show();

}

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

{

this.Hide();

var Task = new TasksForm();

Task.Closed += (s, args) => this.Close();

Task.Show();

}

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

{

this.Hide();

var emailf = new EmailForm();

emailf.Closed += (s, args) => this.Close();

emailf.Show();

}

}

}

# **Email Form**

//Used this tutorial to create this form http://foxlearn.com/article/send-and-receive-email-in-microsoft-outlook-using-csharp-309.html

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 System.Net.Mail;

using Outlook = Microsoft.Office.Interop.Outlook;

namespace Insert

{

public partial class EmailForm : Form

{

public EmailForm()

{

InitializeComponent();

}

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

{

try

{

SmtpClient client = new SmtpClient("smtp.gmail.com", 587);

MailMessage mail = new MailMessage();

mail.From = new MailAddress(Receivertxt.Text);

mail.To.Add(Receivertxt.Text);

mail.Body = Bodytxt.Text;

mail.Subject = SubjectTxt.Text;

if (Attachmenttxt.Text != null)

{

mail.Attachments.Add(new Attachment(Attachmenttxt.Text));

}

client.Credentials = new System.Net.NetworkCredential(emailtxt.Text, passtxt.Text);

client.Send(mail); //Sending Email

MessageBox.Show("Email has been successfully sent.");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, " Message unable to send", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

DataTable DataTbl;

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

{

try

{

Outlook.\_Application OutlookApp = new Outlook.Application();

Outlook.\_NameSpace Name\_space = OutlookApp.GetNamespace("MAPI");

Outlook.MAPIFolder inbox = Name\_space.GetDefaultFolder(Outlook.OlDefaultFolders.olFolderInbox);

Name\_space.SendAndReceive(true);

DataTbl = new DataTable("Inbox");

DataTbl.Columns.Add("Subject", typeof(String));

DataTbl.Columns.Add("Sender", typeof(String));

DataTbl.Columns.Add("Body", typeof(String));

DataTbl.Columns.Add("Date", typeof(String));

dataGridView1.DataSource = DataTbl;

foreach (Outlook.MailItem item in inbox.Items)

DataTbl.Rows.Add(new object[] { item.Subject, item.SenderName, item.HTMLBody, item.SentOn.ToLongDateString() + " " + item.SentOn.ToLongTimeString() });

}

catch (Exception ex)

{

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

}

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex< DataTbl.Rows.Count && e.RowIndex >= 0)

webBrowser1.DocumentText = DataTbl.Rows[e.RowIndex]["Body"].ToString();

}

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

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

Attachmenttxt.Text = openFileDialog1.FileName.ToString();

}

}

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

{

try

{

Outlook.\_Application OutlookApp = new Outlook.Application();

Outlook.\_NameSpace Name\_space = OutlookApp.GetNamespace("MAPI");

Outlook.MAPIFolder inbox = Name\_space.GetDefaultFolder(Outlook.OlDefaultFolders.olFolderInbox);

Name\_space.SendAndReceive(true);

DataTbl = new DataTable("Inbox");

DataTbl.Columns.Add("Subject", typeof(string));

DataTbl.Columns.Add("Sender", typeof(string));

DataTbl.Columns.Add("Body", typeof(string));

DataTbl.Columns.Add("Date", typeof(string));

dataGridView1.DataSource = DataTbl;

foreach (Outlook.MailItem item in inbox.Items)

DataTbl.Rows.Add(new object[] { item.Subject, item.SenderName, item.HTMLBody, item.SentOn.ToLongDateString() + " " + item.SentOn.ToLongTimeString() });

}

catch (Exception ex)

{

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

}

}

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

{

this.Hide();

var dashb = new Dashboard();

dashb.FormClosed += (s, args) => this.Close();

dashb.Show();

}

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

{

this.Hide();

var contact = new ContactsForm();

contact.Closed += (s, args) => this.Close();

contact.Show();

}

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

{

this.Hide();

var cal = new Calendar();

cal.Closed += (s, args) => this.Close();

cal.Show();

}

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

{

this.Hide();

var Task = new TasksForm();

Task.Closed += (s, args) => this.Close();

Task.Show();

}

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

{

this.Hide();

var Invent = new InventManagement();

Invent.Closed += (s, args) => this.Close();

Invent.Show();

}

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

{

this.Hide();

var Cust = new CustomersForm();

Cust.Closed += (s, args) => this.Close();

Cust.Show();

}

}

}

# **Inventory Management Form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.OleDb;

using System.IO;

namespace Insert

{

public partial class InventManagement : Form

{

OleDbConnection cnon = new OleDbConnection();

private static string json\_secret\_file = @".\client\_secret.json";

private static string application\_name = @".GoogleDriveUploader";

private string GetFileType(string file)

{

string extension = Path.GetExtension(file);

System.Diagnostics.Debug.WriteLine("extension: " + extension);

string mime;

switch (extension.ToLower())

{

case ".jpg":

mime = "image/jpeg";

break;

case ".jpeg":

mime = "image/jpeg";

break;

case ".png":

mime = "image/png";

break;

case ".accdb":

mime = "application/msaccess";

break;

case ".pptx":

mime = "application/vnd.google-apps.presentation";

break;

case ".xlsx":

mime = "application/vnd.google-apps.spreadsheet";

break;

case ".doc":

mime = "application/vnd.google-apps.document";

break;

case ".docx":

mime = "application/vnd.google-apps.document";

break;

case ".pdf":

mime = "application/pdf";

break;

default:

mime = "text/plain";

break;

}

return mime;

}

public InventManagement()

{

InitializeComponent();

cnon.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=|DataDirectory|\Contacts.accdb";

}

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

{

string query = "SELECT \* From Inventory";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

InventoryGridList.DataSource = ds.Tables[0];

}

}

private void InventoryGridList\_CellContentClick\_1(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0)

{

//gets a collection that contains all the rows

DataGridViewRow row = this.InventoryGridList.Rows[e.RowIndex];

//populate the textbox from specific value of the coordinates of column and row.

InventIDtxt.Text = row.Cells[0].Value.ToString();

serialNoTxt.Text = row.Cells[1].Value.ToString();

itemNametxt.Text = row.Cells[2].Value.ToString();

clientComptxt.Text = row.Cells[3].Value.ToString();

orderID.Text = row.Cells[4].Value.ToString();

OrderDTP.Text = row.Cells[5].Value.ToString();

ShipDTP.Text = row.Cells[6].Value.ToString();

modelNoTxt.Text = row.Cells[7].Value.ToString();

Typetxt.Text = row.Cells[8].Value.ToString();

MaintDTP.Text = row.Cells[9].Value.ToString();

Pricetxt.Text = row.Cells[10].Value.ToString();

}

}

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

{

serialNoTxt.Enabled = true;

itemNametxt.Enabled = true;

clientComptxt.Enabled = true;

clientComptxt.Enabled = true;

orderID.Enabled = true;

OrderDTP.Enabled = true;

ShipDTP.Enabled = true;

modelNoTxt.Enabled = true;

Typetxt.Enabled = true;

MaintDTP.Enabled = true;

Pricetxt.Enabled = true;

SaveBtn.Enabled = true;

EditBtn.Enabled = false;

DeleteBtn.Enabled = false;

serialNoTxt.Text = "";

itemNametxt.Text = "";

clientComptxt.Text = "";

orderID.Text = "";

modelNoTxt.Text = "";

Typetxt.Text = "";

Pricetxt.Text = "";

}

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

{

if (serialNoTxt.Text == "" | itemNametxt.Text == "" | clientComptxt.Text == "" | orderID.Text == "" | modelNoTxt.Text == "" | Typetxt.Text == "" | Pricetxt.Text == "")

{

MessageBox.Show("All text boxes must be filled");

}

else

{

OleDbCommand command = new OleDbCommand("INSERT INTO Inventory (SerialNo,ItemName, ClientCompany, OrderID, DateOrder, DateShipped, ModelNo, Type, MaintenanceDate, Price) VALUES(?, ?, ?, ?, ?, ? ,?, ?, ?, ?)", cnon);

command.Parameters.AddWithValue("@SerialNo", serialNoTxt.Text);

command.Parameters.AddWithValue("@ItemName", itemNametxt.Text);

command.Parameters.AddWithValue("@ClientCompany", clientComptxt.Text);

command.Parameters.AddWithValue("@OrderID", orderID.Text);

command.Parameters.AddWithValue("@DateOrder", OrderDTP.Value.ToString());

command.Parameters.AddWithValue("@DateShipped", ShipDTP.Value.ToString());

command.Parameters.AddWithValue("@ModelNo", modelNoTxt.Text);

command.Parameters.AddWithValue("@Type", Typetxt.Text);

command.Parameters.AddWithValue("@MaintenanceDate", MaintDTP.Value.ToString());

command.Parameters.AddWithValue("@Price", Pricetxt.Text);

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("WELL DONE");

cnon.Close();

//

//

//

serialNoTxt.Text = "";

itemNametxt.Text = "";

clientComptxt.Text = "";

orderID.Text = "";

modelNoTxt.Text = "";

Typetxt.Text = "";

Pricetxt.Text = "";

}

//

//Adds the new record into the datagridview by loading the database again.

//

string query = "SELECT \* From Inventory";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

InventoryGridList.DataSource = ds.Tables[0];

}

//

//

//

}

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

{

if (serialNoTxt.Text == "" | itemNametxt.Text == "" | clientComptxt.Text == "" | orderID.Text == "" | modelNoTxt.Text == "" | Typetxt.Text == "" | Pricetxt.Text == "")

{

MessageBox.Show("All text boxes must be filled");

}

else

{

string sqlquery = "UPDATE Inventory SET SerialNo = ?, ItemName = ?, ClientCompany = ?, OrderID = ?, DateOrder = ?, DateShipped = ?, ModelNo = ?, Type = ?, MaintenanceDate = ?, Price = ? WHERE InventNo = " + InventIDtxt.Text + "";

using (OleDbCommand command = new OleDbCommand(sqlquery, cnon))

{

command.Parameters.AddWithValue("@SerialNo", serialNoTxt.Text);

command.Parameters.AddWithValue("@ItemName", itemNametxt.Text);

command.Parameters.AddWithValue("@ClientCompany", clientComptxt.Text);

command.Parameters.AddWithValue("@OrderID", orderID.Text);

command.Parameters.AddWithValue("@DateOrder", OrderDTP.Value.ToString());

command.Parameters.AddWithValue("@DateShipped", ShipDTP.Value.ToString());

command.Parameters.AddWithValue("@ModelNo", modelNoTxt.Text);

command.Parameters.AddWithValue("@Type", Typetxt.Text);

command.Parameters.AddWithValue("@MaintenanceDate", MaintDTP.Value.ToString());

command.Parameters.AddWithValue("@Price", Pricetxt.Text);

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("WELL DONE");

cnon.Close();

//

//

//

serialNoTxt.Enabled = false;

itemNametxt.Enabled = false;

clientComptxt.Enabled = false;

clientComptxt.Enabled = false;

orderID.Enabled = false;

OrderDTP.Enabled = false;

ShipDTP.Enabled = false;

modelNoTxt.Enabled = false;

Typetxt.Enabled = false;

MaintDTP.Enabled = false;

Pricetxt.Enabled = false;

SaveBtn.Enabled = false;

AddBtn.Enabled = true;

UpdateBtn.Enabled = false;

EditBtn.Enabled = true;

DeleteBtn.Enabled = true;

serialNoTxt.Text = "";

itemNametxt.Text = "";

clientComptxt.Text = "";

orderID.Text = "";

modelNoTxt.Text = "";

Typetxt.Text = "";

Pricetxt.Text = "";

}

//

//Adds the new record into the datagridview by loading the database again.

//

string query = "SELECT \* From Inventory";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

InventoryGridList.DataSource = ds.Tables[0];

}

//

//

//

}

}

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

{

if ((e.KeyChar == '.') && ((sender as TextBox).Text.IndexOf('.') > -1))

{

e.Handled = true;

}

}

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

{

SaveBtn.Enabled = false;

AddBtn.Enabled = true;

UpdateBtn.Enabled = true;

EditBtn.Enabled = true;

DeleteBtn.Enabled = true;

serialNoTxt.Enabled = false;

itemNametxt.Enabled = false;

clientComptxt.Enabled = false;

clientComptxt.Enabled = false;

orderID.Enabled = false;

OrderDTP.Enabled = false;

ShipDTP.Enabled = false;

modelNoTxt.Enabled = false;

Typetxt.Enabled = false;

MaintDTP.Enabled = false;

Pricetxt.Enabled = false;

}

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

{

serialNoTxt.Enabled = true;

itemNametxt.Enabled = true;

clientComptxt.Enabled = true;

clientComptxt.Enabled = true;

orderID.Enabled = true;

OrderDTP.Enabled = true;

ShipDTP.Enabled = true;

modelNoTxt.Enabled = true;

Typetxt.Enabled = true;

MaintDTP.Enabled = true;

Pricetxt.Enabled = true;

SaveBtn.Enabled = false;

AddBtn.Enabled = false;

UpdateBtn.Enabled = true;

DeleteBtn.Enabled = false;

}

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

{

serialNoTxt.Enabled = false;

itemNametxt.Enabled = false;

clientComptxt.Enabled = false;

clientComptxt.Enabled = false;

orderID.Enabled = false;

OrderDTP.Enabled = false;

ShipDTP.Enabled = false;

modelNoTxt.Enabled = false;

Typetxt.Enabled = false;

MaintDTP.Enabled = false;

Pricetxt.Enabled = false;

SaveBtn.Enabled = false;

AddBtn.Enabled = true;

UpdateBtn.Enabled = false;

EditBtn.Enabled = true;

DeleteBtn.Enabled = true;

serialNoTxt.Text = "";

itemNametxt.Text = "";

clientComptxt.Text = "";

orderID.Text = "";

modelNoTxt.Text = "";

Typetxt.Text = "";

Pricetxt.Text = "";

}

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

{

sync();

this.Hide();

var dashb = new Dashboard();

dashb.Closed += (s, args) => this.Close();

dashb.Show();

}

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

{

sync();

this.Hide();

var cal = new Calendar();

cal.Closed += (s, args) => this.Close();

cal.Show();

}

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

{

sync();

this.Hide();

var contact = new ContactsForm();

contact.Closed += (s, args) => this.Close();

contact.Show();

}

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

{

sync();

this.Hide();

var Task = new TasksForm();

Task.Closed += (s, args) => this.Close();

Task.Show();

}

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

{

sync();

this.Hide();

var emailf = new EmailForm();

emailf.Closed += (s, args) => this.Close();

emailf.Show();

}

public void sync()

{

string pathfile = Path.GetDirectoryName(Application.ExecutablePath) + @"\Contacts.accdb";

string filename = "Contacts.accdb";

CloudUpload driveUploader = new CloudUpload(json\_secret\_file, application\_name);

System.Diagnostics.Debug.WriteLine(pathfile);

try

{

byte[] byteArray = System.IO.File.ReadAllBytes(pathfile);

string filePath = pathfile;

string fileName = filename;

string fileId = "1rRvc7aKnzKVvvTBwtOnT1-1GO2kz6iK9";

string description = "test";

string fileType = GetFileType(filePath);

driveUploader.UploadFile(ref byteArray, fileName, fileId, description, fileType);

}

catch (Exception exc)

{

System.Diagnostics.Debug.WriteLine(exc.Message);

}

}

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

{

this.Hide();

var Cust = new CustomersForm();

Cust.Closed += (s, args) => this.Close();

Cust.Show();

}

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

{

string query = "DELETE from Inventory WHERE InventNo=" + InventIDtxt.Text;

using (OleDbCommand command = new OleDbCommand(query, cnon))

{

command.CommandType = CommandType.Text;

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("Delete Successful");

cnon.Close();

}

serialNoTxt.Text = "";

itemNametxt.Text = "";

clientComptxt.Text = "";

orderID.Text = "";

modelNoTxt.Text = "";

Typetxt.Text = "";

Pricetxt.Text = "";

string query2 = "SELECT \* From Inventory";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query2, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

InventoryGridList.DataSource = ds.Tables[0];

}

}

}

}

# **Mime Type Converter**

// Use this tutorial to help implement google drive https://www.daimto.com/google-drive-api-c-upload/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Insert

{

public class MimeTypeConvert

{

public string extension { get; set; }

public string mimeType { get; set; }

public string converterType { get; set; }

public MimeTypeConvert()

{

}

public MimeTypeConvert(string extension, string type, string converter)

{

this.extension = extension;

mimeType = type;

converterType = converter;

}

}

public static class MimeConverter

{

public static List<MimeTypeConvert> mimeList()

{

List<MimeTypeConvert> list = new List<MimeTypeConvert>();

list.Add(new MimeTypeConvert(".xlsx", "application/vnd.google-apps.spreadsheet", "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"));

list.Add(new MimeTypeConvert(".doc", "application/vnd.google-apps.document", "application/vnd.openxmlformats-officedocument.wordprocessingml.document"));

list.Add(new MimeTypeConvert(".pptx", "application/vnd.google-apps.presentation", "application/vnd.openxmlformats-officedocument.presentationml.presentation"));

list.Add(new MimeTypeConvert(".html", "application/vnd.google-apps.site", "text/html"));

return list;

}

}

}

# **Tasks Form**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.OleDb;

using System.IO;

namespace Insert

{

public partial class TasksForm : Form

{

//

//Establish the database connection

//

OleDbConnection cnon = new OleDbConnection();

//

//

//

private static string json\_secret\_file = @".\client\_secret.json";

private static string application\_name = @".GoogleDriveUploader";

private string GetFileType(string file)

{

string extension = Path.GetExtension(file);

System.Diagnostics.Debug.WriteLine("extension: " + extension);

string mime;

switch (extension.ToLower())

{

case ".jpg":

mime = "image/jpeg";

break;

case ".jpeg":

mime = "image/jpeg";

break;

case ".png":

mime = "image/png";

break;

case ".accdb":

mime = "application/msaccess";

break;

case ".pptx":

mime = "application/vnd.google-apps.presentation";

break;

case ".xlsx":

mime = "application/vnd.google-apps.spreadsheet";

break;

case ".doc":

mime = "application/vnd.google-apps.document";

break;

case ".docx":

mime = "application/vnd.google-apps.document";

break;

case ".pdf":

mime = "application/pdf";

break;

default:

mime = "text/plain";

break;

}

return mime;

}

public TasksForm()

{

InitializeComponent();

//Pathway to the database

//

cnon.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=|DataDirectory|\Contacts.accdb";

//

//

// To display single selected of date

monthCalendar1.MaxSelectionCount = 1;

}

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

{

if (TaskNameTB.Text == "")

{

MessageBox.Show("Please fill in Task Name textbox");

}

else

{

OleDbCommand command = new OleDbCommand("INSERT INTO Tasks (TaskName, TaskDesc, StartDate, DueDate) VALUES(?, ? ,?, ?)", cnon);

command.Parameters.AddWithValue("@TaskName", TaskNameTB.Text);

command.Parameters.AddWithValue("@TaskDesc", TaskDescTB.Text);

command.Parameters.AddWithValue("@StartDate", StartDateDTP.Value.ToString());

command.Parameters.AddWithValue("@DueDate", DueDateDTP.Value.ToShortDateString());

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("WELL DONE");

cnon.Close();

SaveBtn.Enabled = false;

AddBtn.Enabled = true;

UpdateBtn.Enabled = false;

EditBtn.Enabled = true;

DeleteBtn.Enabled = true;

TaskNameTB.Enabled = false;

TaskDescTB.Enabled = false;

StartDateDTP.Enabled = false;

DueDateDTP.Enabled = false;

IDTB.Text = "";

TaskNameTB.Text = "";

TaskDescTB.Text = "";

//

//Adds the new record into the datagridview by loading the database again.

//

string query = "SELECT \* From Tasks";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

SelectedDateDGV.DataSource = ds.Tables[0];

}

//

//

//

}

}

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

{

if (TaskNameTB.Text == "")

{

MessageBox.Show("Please fill in all boxes");

}

else

{

string sqlquery = "UPDATE Tasks SET TaskName = ?, TaskDesc = ?, StartDate = ?, DueDate = ? WHERE TaskID = " + IDTB.Text + "";

using (OleDbCommand command = new OleDbCommand(sqlquery, cnon))

{

command.Parameters.AddWithValue("@TaskName", TaskNameTB.Text);

command.Parameters.AddWithValue("@TaskDesc", TaskDescTB.Text);

command.Parameters.AddWithValue("@StartDate", StartDateDTP.Value.ToString());

command.Parameters.AddWithValue("@DueDate", DueDateDTP.Value.ToShortDateString());

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("Update Successful");

cnon.Close();

}

IDTB.Text = "";

TaskNameTB.Text = "";

TaskDescTB.Text = "";

AddBtn.Enabled = true;

DeleteBtn.Enabled = true;

UpdateBtn.Enabled = false;

//

//Adds the new record into the datagridview by loading the database again.

//

string query = "SELECT \* From Tasks";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

SelectedDateDGV.DataSource = ds.Tables[0];

}

//

//

//

}

}

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

{

SaveBtn.Enabled = true;

EditBtn.Enabled = false;

DeleteBtn.Enabled = false;

TaskNameTB.Enabled = true;

TaskDescTB.Enabled = true;

StartDateDTP.Enabled = true;

DueDateDTP.Enabled = true;

IDTB.Text = "";

TaskNameTB.Text = "";

TaskDescTB.Text = "";

}

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

{

AddBtn.Enabled = false;

DeleteBtn.Enabled = false;

UpdateBtn.Enabled = true;

TaskNameTB.Enabled = true;

TaskDescTB.Enabled = true;

StartDateDTP.Enabled = true;

DueDateDTP.Enabled = true;

}

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

{

if (IDTB.Text == "")

{

MessageBox.Show("Select a record)");

}

else

{

string query = "DELETE from Tasks WHERE TaskID=" + IDTB.Text;

using (OleDbCommand command = new OleDbCommand(query, cnon))

{

command.CommandType = CommandType.Text;

cnon.Open();

command.Connection = cnon;

command.ExecuteNonQuery();

MessageBox.Show("Delete Successful");

cnon.Close();

}

IDTB.Text = "";

TaskNameTB.Text = "";

TaskDescTB.Text = "";

string query2 = "SELECT \* From Tasks";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query2, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

SelectedDateDGV.DataSource = ds.Tables[0];

}

}

}

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

{

}

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

{

string query = "SELECT \* From Tasks";

using (OleDbDataAdapter adapter = new OleDbDataAdapter(query, cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

SelectedDateDGV.DataSource = ds.Tables[0];

}

}

private void monthCalendar1\_DateChanged(object sender, DateRangeEventArgs e)

{

// To display single selected of date

monthCalendar1.MaxSelectionCount = 1;

// To display single date use MonthCalendar1.SelectionRange.Start/ MonthCalendarSelectionRange.End

SelectedDateTB.Text = monthCalendar1.SelectionRange.End.ToShortDateString();

//string query = "SELECT \* From Events WHERE EventDate = " + SelectedDateTB.Text;

using (OleDbDataAdapter adapter = new OleDbDataAdapter("SELECT \* From Tasks WHERE DueDate =#" + SelectedDateTB.Text + "#", cnon))

{

DataSet ds = new DataSet();

adapter.Fill(ds);

SelectedDateDGV.DataSource = ds.Tables[0];

}

}

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

{

if (e.RowIndex >= 0)

{

//gets a collection that contains all the rows

DataGridViewRow row = this.SelectedDateDGV.Rows[e.RowIndex];

//populate the textbox from specific value of the coordinates of column and row.

IDTB.Text = row.Cells[0].Value.ToString();

TaskNameTB.Text = row.Cells[1].Value.ToString();

TaskDescTB.Text = row.Cells[2].Value.ToString();

StartDateDTP.Text = row.Cells[3].Value.ToString();

DueDateDTP.Text = row.Cells[4].Value.ToString();

}

}

public void sync()

{

string pathfile = Path.GetDirectoryName(Application.ExecutablePath) + @"\Contacts.accdb";

string filename = "Contacts.accdb";

CloudUpload driveUploader = new CloudUpload(json\_secret\_file, application\_name);

System.Diagnostics.Debug.WriteLine(pathfile);

try

{

byte[] byteArray = System.IO.File.ReadAllBytes(pathfile);

string filePath = pathfile;

string fileName = filename;

string fileId = "1rRvc7aKnzKVvvTBwtOnT1-1GO2kz6iK9";

string description = "test";

string fileType = GetFileType(filePath);

driveUploader.UploadFile(ref byteArray, fileName, fileId, description, fileType);

}

catch (Exception exc)

{

System.Diagnostics.Debug.WriteLine(exc.Message);

}

}

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

{

sync();

this.Hide();

var dashb = new Dashboard();

dashb.Closed += (s, args) => this.Close();

dashb.Show();

}

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

{

sync();

this.Hide();

var contact = new ContactsForm();

contact.Closed += (s, args) => this.Close();

contact.Show();

}

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

{

sync();

this.Hide();

var cal = new Calendar();

cal.Closed += (s, args) => this.Close();

cal.Show();

}

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

{

sync();

this.Hide();

var Invent = new InventManagement();

Invent.Closed += (s, args) => this.Close();

Invent.Show();

}

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

{

sync();

this.Hide();

var emailf = new EmailForm();

emailf.Closed += (s, args) => this.Close();

emailf.Show();

}

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

{

this.Hide();

var Cust = new CustomersForm();

Cust.Closed += (s, args) => this.Close();

Cust.Show();

}

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

{

}

}

}