using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Forms;

using System.Xml.Linq;

namespace RegoReader\_v1.\_0

{

public partial class RegistrationReaderForm : Form

{

public RegistrationReaderForm()

{

InitializeComponent();

}

//

// global variables.

//

public bool enableDoubleClick = true;

public String[] regoArray;

//

// Open button click causes an open file dialog to open.

// Filters for txt files

//

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

{

OpenFileDialog openFileDialog1 = new OpenFileDialog

{

InitialDirectory = @"D:\",

Title = "Browse Text Files",

CheckFileExists = true,

CheckPathExists = true,

DefaultExt = "txt",

Filter = "txt files (\*.txt)|\*.txt",

FilterIndex = 1,

RestoreDirectory = true,

ReadOnlyChecked = true,

ShowReadOnly = true

};

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

{

string filename = openFileDialog1.FileName;

regoArray = File.ReadAllLines(filename);

int regoArrayLength = regoArray.Length;

DisplayArray(regoArray, regoArrayLength);

TextBoxErrorMes.Text = "File Opened";

}

}

//

// Add Button:

// Will add entered text into listbox - auto capitalised.

// Check for valid entry - if not, will display an error message.

//

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

{

ListBoxRego.Update();

if ((!ListBoxRego.Items.Contains(TextBoxEntry.Text)) && (!string.IsNullOrWhiteSpace(TextBoxEntry.Text)))

{

ListBoxRego.Items.Add(TextBoxEntry.Text);

TextBoxEntry.Clear();

TextBoxErrorMes.Text = "Entry Added";

}

else

{

TextBoxEntry.Clear();

TextBoxErrorMes.Text = "Enter a Valid Registration Plate";

}

}

//

// DisplayArray method to write array elements into listbox.

// Checks for duplicate on each element and skips it if element is

// a duplicate.

//

private void DisplayArray(string[] array, int length)

{

for (int i = 0; i < length; i++)

{

if (!ListBoxRego.Items.Contains(array[i]))

{

ListBoxRego.Items.Add(array[i]);

}

else

{

continue;

}

}

}

//

//Delete item with double click

//

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

{

if (ListBoxRego.SelectedItem != null &&

enableDoubleClick == true)

{

var confirm = MessageBox.Show("Are you sure you want to delete this registration plate?", "Delete Confirmation", MessageBoxButtons.YesNo);

if ((confirm == DialogResult.Yes))

{

ListBoxRego.Items.Remove(ListBoxRego.SelectedItems[0]);

ListBoxRego.ClearSelected();

TextBoxErrorMes.Text = "Registration plate entry deleted";

}

else

{

ListBoxRego.ClearSelected();

}

}

}

//

// Delete Item with button

//

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

{

if (ListBoxRego.SelectedItem != null)

{

var confirm = MessageBox.Show("Are you sure you want to delete this item?", "Delete Confirmation", MessageBoxButtons.YesNo);

if ((confirm == DialogResult.Yes))

{

ListBoxRego.Items.Remove(ListBoxRego.SelectedItems[0]);

ListBoxRego.ClearSelected();

TextBoxErrorMes.Text = "Registration plate entry deleted";

}

else

{

ListBoxRego.ClearSelected();

}

}

}

//

// Reset button. Clears ListBox.

//

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

{

var confirm = MessageBox.Show("Are you sure you want to reset the list?\n\nAll current information will be lost.", "Reset Confirmation", MessageBoxButtons.YesNo);

if (confirm == DialogResult.Yes)

{

ListBoxRego.Items.Clear();

ClearSelect();

TextBoxErrorMes.Text = "Registratoin plate list cleared";

}

}

//

// Clicking a rego number in the

//

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

{

TextBoxEntry.Text = (string)ListBoxRego.SelectedItem;

ButtonAdd.Enabled = false;

}

//

// Method clears list box selection and re-enables add button

// function

//

private void ClearSelect()

{

ListBoxRego.ClearSelected();

enableDoubleClick = true;

ButtonAdd.Enabled = true;

ButtonDelete.Enabled = true;

}

//

// Edit button. Modifies current selected item with text from text

// box when clicked.

//

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

{

if (ListBoxRego.SelectedItem != null)

{

ListBoxRego.Items[ListBoxRego.SelectedIndex] = TextBoxEntry.Text;

ListBoxRego.Sorted = false;

ListBoxRego.Sorted = true;

ClearSelect();

TextBoxErrorMes.Text = "Plate modified";

}

else

{

TextBoxErrorMes.Text = "No selection made";

}

}

//

// Clicking outside of window elements clears list box selection.

//

private void splitContainer1\_Panel2\_Click(object sender, EventArgs e)

{

ClearSelect();

}

//

// Calls linear search to search rego plates currently stored.

//

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

{

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

{

List<string> templist = MakeTempList();

LinearSearch(templist, TextBoxEntry.Text);

ClearSelect();

TextBoxErrorMes.Text = "Linear search complete";

}

else

{

TextBoxErrorMes.Text = "Enter valid search entry";

}

}

//

// Makes a temporary sorted list to make searches easier to code.

//

private List<string> MakeTempList()

{

List<string> templist = new List<string>();

templist.Clear();

foreach (var item in ListBoxRego.Items)

{

templist.Add((string)item);

}

return templist;

}

//

// Linear Search Method.

//

private void LinearSearch(List<string> list, string key)

{

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

{

if ((list[i] == key) || (list[i] == ("z"+key)))

{

MessageBox.Show("Registration plate exists.","Search Result");

return;

}

}

MessageBox.Show("Registraition plate does not exist.", "Search Result");

return;

}

//

// Binary Search Button. Sorts list then uses built in binary

// search to find rego plate.

//

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

{

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

{

List<string> templist = MakeTempList();

templist.Sort();

if (!(templist.BinarySearch(TextBoxEntry.Text) < 0) || !(templist.BinarySearch("z" + TextBoxEntry.Text) < 0))

{

MessageBox.Show("Registration plate exists.", "Search Result");

}

else

{

MessageBox.Show("Registration plate does not exist.", "Search Result");

}

TextBoxErrorMes.Text = "Binary search complete";

}

else

{

TextBoxErrorMes.Text = "Enter valid search entry";

}

}

//

// Save Button. Calls save file Method.

//

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

{

SaveFile();

}

//

// Save File Method. Creates save file dialogue, and automatically

// increments a suggested name

//

private void SaveFile()

{

SaveFileDialog save = new SaveFileDialog();

save.InitialDirectory = "D:\\DemoRego";

save.Filter = "Text Files (\*.txt)|\*.txt";

save.DefaultExt = ".txt";

save.Title = "Select Save Location";

int num = 0;

string filename;

string extension = ".txt";

string name;

do

{

name = "demo\_";

if (num < 9)

{

name = "demo\_0";

num++;

filename = String.Format("{0}{1}", name, num);

}

else

{

num++;

filename = String.Format("{0}{1}", name, num);

}

} while (File.Exists("D:\\DemoRego\\"+filename+extension));

// do-while from: https://stackoverflow.com/questions/10131667/automatically-increment-filename

save.FileName = filename;

DialogResult result = save.ShowDialog();

// checks if file name is valid and "save" is pressed in savedialog.

if ((save.FileName != "") && (result == DialogResult.OK))

{

System.IO.StreamWriter sw = new StreamWriter("D:\\DemoRego\\" + filename+extension);

foreach (var item in ListBoxRego.Items)

{

sw.WriteLine(item.ToString());

}

TextBoxErrorMes.Text = String.Format("File saved as: {0}{1}{2}", name, num, extension);

sw.Close();

// Write to file From: https://stackoverflow.com/questions/20595279/c-sharp-save-all-items-in-a-listbox-to-text-file

}

}

//

// Exit Button. Confirms exit of program and Calls the save file

// method before closing.

//

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

{

var result = MessageBox.Show("Are you sure you want to exit?\n(Program will save before closing)", "Exit Confirmation", MessageBoxButtons.YesNo);

if (result == DialogResult.Yes)

{

SaveFile();

Close();

}

}

//

// Tab Button. Will add a "z" to the front of a rego plate and

// remove it if it starts with a z

//

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

{

if (ListBoxRego.SelectedIndex != -1)

{

string temp = ListBoxRego.SelectedItem.ToString();

if (temp.Substring(0, 1) != "z")

{

ListBoxRego.Items[ListBoxRego.SelectedIndex] = "z" + temp;

ClearSelect();

ListBoxRego.Sorted = false;

ListBoxRego.Sorted = true;

TextBoxErrorMes.Text = "Registration entry: " + temp + " tagged";

}

// From:https://www.tutorialspoint.com/How-to-find-the-first-character-of-a-string-in-Chash#:~:text=How%20to%20find%20the%20first%20character%20of%20a%20string%20in%20C%23%3F&text=To%20get%20the%20first%20character%2C%20use%20the%20substring()%20method.&text=string%20str%20%3D%20%22Welcome%20to%20the,in%20the%20substring()%20method.

else

{

TextBoxErrorMes.Text = "Registration entry: " + temp +" untagged";

temp = temp.Remove(0, 1);

ListBoxRego.Items[ListBoxRego.SelectedIndex] = temp;

ClearSelect();

ListBoxRego.Sorted = false;

ListBoxRego.Sorted = true;

}

//From:https://stackoverflow.com/questions/3222125/fastest-way-to-remove-first-char-in-a-string

}

}

//

// All Tooltip hovers. Give breif descripton of each element

// function.

// Code based off:https://www.youtube.com/watch?v=izv5S38ctBs

//

private void TextBoxEntry\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Enter Registraion Plate Information Here", TextBoxEntry);

}

private void ButtonAdd\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Adds text in entry field to list", ButtonAdd);

}

private void ButtonEdit\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Replaces selection with text in entry field", ButtonEdit);

}

private void ButtonDelete\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Deletes selection", ButtonDelete);

}

private void ButtonOpen\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Browse text files and import contents", ButtonOpen);

}

private void ButtonSave\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Saves content of list to text file", ButtonSave);

}

private void ButtonExit\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Save and Exit Program", ButtonExit);

}

private void ListBoxRego\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("List of Registration Plates\n\nDouble-Click Item to Delete", ListBoxRego);

}

private void ButtonTag\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Add a \"z\" to the front of selected plate number\n\nPress \"Tag\" again to remove.", ButtonTag);

}

private void ButtonReset\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Clears the list\n\nList will NOT save", ButtonReset);

}

private void ButtonLinSearch\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Check if plate exists in list\n\nsearches using linear search", ButtonLinSearch);

}

private void ButtonBinSearch\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Check if plate exists in list\n\nsearches using binary search", ButtonBinSearch);

}

}

}