```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using System.Windows.Forms;
6
7  namespace SKOffice
8  {
9      static class Program
10     {
11         /// <summary>
12         /// The main entry point for the application.
13         /// </summary>
14         [STAThread]
15         static void Main()
16         {
17             Application.EnableVisualStyles();
18             Application.SetCompatibleTextRenderingDefault(false);
19             Application.Run(new MainForm());
20         }
21     }
22 }
23
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Threading;
5  using System.Windows.Forms;
6  using WcfService.domain.order;
7
8  namespace SKOffice
9  {
10     public partial class MainForm : Form
11     {
12         OpenFileDialog ofd;
13         public string[] paths;
14         private List<OrderConfirmation> orderConfirmations;
15         private int updates;
16         private Thread updateThread;
17         private int updateFrequency;
18
19         /// <summary>
20         /// Instantiates the listview including the columns.
21         /// Creating and starting the thread that will auto update the list
22         /// Instantiating variables
23         /// </summary>
24         public MainForm()
25         {
26             updates = 0;
27             updateThread = new Thread(new ThreadStart(updateLoop));
28             updateFrequency = 30000;
29             updateThread.Start();
30             paths = new string[3];
31             InitializeComponent();
32
33             // Set the view to show details.
34             orderOverViewList.View = View.Details;
35
36             // Display grid lines.
37             orderOverViewList.GridLines = true;
38
39             // Create columns for the items and subitems.
40             // Width of -2 indicates auto-size.
41             orderOverViewList.Columns.Add("Order ID", 150,
                 HorizontalAlignment.Center);
42             orderOverViewList.Columns.Add("Started", 50,
                 HorizontalAlignment.Center);
43             orderOverViewList.Columns.Add("Done", 50,
                 HorizontalAlignment.Center);
44             orderOverViewList.Columns.Add("Started", 50,
                 HorizontalAlignment.Center);
45             orderOverViewList.Columns.Add("Done", 50,
                 HorizontalAlignment.Center);
46             orderOverViewList.Columns.Add("Started", 50,
                 HorizontalAlignment.Center);
47             orderOverViewList.Columns.Add("Done", 50,
                 HorizontalAlignment.Center);
48             orderOverViewList.Columns.Add("Started", 50,
                 HorizontalAlignment.Center);
```

```
49              orderOverViewList.Columns.Add("Done", 50,
                  HorizontalAlignment.Center);
50              orderOverViewList.Columns.Add("Started", 50,
                  HorizontalAlignment.Center);
51              orderOverViewList.Columns.Add("Done", 50,
                  HorizontalAlignment.Center);
52              orderOverViewList.Columns.Add("Note", -2,
                  HorizontalAlignment.Left);
53          }
54
55          private void MainForm_Load(object sender, EventArgs e)
56          {
57          }
58
59          /// <summary>
60          /// When the tabpage are clicked on, the list is updated
61          /// </summary>
62          /// <param name="sender"></param>
63          /// <param name="e"></param>
64          private void tabPage1_Click(object sender, EventArgs e)
65          {
66              updateList();
67          }
68
69
70          /// <summary>
71          /// When one of the browse buttons are clicked the user are prompted
              to browse for the e02 file.
72          /// </summary>
73          /// <param name="sender"></param>
74          /// <param name="e"></param>
75          private void browse_Click(object sender, EventArgs e)
76          {
77              ofd = new OpenFileDialog();
78
79              //fbd.RootFolder = Environment.SpecialFolder.Desktop;
80
81              Button btn = (Button)sender;
82              switch (btn.Name)
83              {
84                  case "browseE02Btn":
85                      ofd.Filter = "Text Files (.e02)|*.e02";
86                      ofd.FilterIndex = 1;
87                      ofd.Multiselect = false;
88                      ofd.ShowDialog();
89                      tb_e02.Text = ofd.FileName;
90                      paths[0] = tb_e02.Text;
91                      break;
92                  /*case "browseBlueprintBtn":
93                      ofd.Filter = "Text Files (.pdf)|*.pdf";
94                      ofd.FilterIndex = 1;
95                      ofd.Multiselect = false;
96                      ofd.ShowDialog();
97                      tb_Blueprints.Text = ofd.FileName;
98                      paths.Add(tb_Blueprints.Text);
99                      break;
```

```
100                 case "browseRequisitionBtn":
101                     ofd.Filter = "Text Files (.pdf)|*.pdf";
102                     ofd.FilterIndex = 1;
103                     ofd.Multiselect = false;
104                     ofd.ShowDialog();
105                     tb_Requisition.Text = ofd.FileName;
106                     paths.Add(tb_Requisition.Text);
107                     break;*/
108                 default:
109                     break;
110             }
111         }
112 
113         /// <summary>
114         /// When the Upload button is Clicked, it will take the file from the path and uploads it
115         /// A box will show the status for the upload, if its uploaded Green and failed Red, idle is Blue
116         /// </summary>
117         /// <param name="sender"></param>
118         /// <param name="e"></param>
119         private void uploadBtn_Click(object sender, EventArgs e)
120         {
121             feedbackE02.BackColor = System.Drawing.Color.RoyalBlue; // Feedback: idle
122             RestService.RestServiceClient rsClient = new RestService.RestServiceClient();
123 
124             //Read the content of the file into a string array
125             List<string> fileContent = new List<string>();
126             FileInfo fileInfo = new FileInfo(paths[0]);
127             fileContent.Add(fileInfo.Name);
128             using (StreamReader sReader = new StreamReader(paths[0]))
129             {
130                 while (sReader.Peek() > -1)
131                     fileContent.Add(sReader.ReadLine());
132             }
133             string msg = rsClient.addOrderConfirmation(fileContent.ToArray());
134             //Tries to add the order confirmation on the service
135             if (msg.StartsWith("OK"))
136             {
137                 feedbackE02.BackColor = System.Drawing.Color.Green; // Feedback: success
138                 tb_e02.Clear();
139                 paths[0] = "";
140             }
141             else
142                 feedbackE02.BackColor = System.Drawing.Color.Red; //Feedback: failed
143             MessageBox.Show(msg , "Service Response");
144         }
145 
146         /// <summary>
147         /// Double clicking on an order number it will open a window
148         /// </summary>
149         /// <param name="sender"></param>
```

```
150         /// <param name="e"></param>
151         private void clickedOrder(object sender, MouseEventArgs e)
152         {
153             int y = 24;
154             for (int i = 0; i < orderOverViewList.Items.Count; i++)
155             {
156                 if (e.Y >= y && e.Y <= y + 16)
157                 {
158                     Console.WriteLine(e.Y + " " + i);
159                     OrderForm orderForm = new OrderForm(orderConfirmations
                        [i]);
160                     orderForm.Show();
161                     break;
162                 }
163                 //each row is 16 pixels in height
164                 //So each element is 0-16, 17-32, 33-48
165                 y += 17;
166             }
167         }
168 
169         private delegate void UniversalVoidDelegate();
170 
171         /// <summary>
172         /// Recieves a call from the form and allows the listview to be
              updated from another thread.
173         /// </summary>
174         /// <param name="control"></param>
175         /// <param name="function"></param>
176         public static void controlInvoke(Control control, Action function)
177         {
178             if (control.IsDisposed || control.Disposing)
179                 return;
180 
181             if (control.InvokeRequired)
182             {
183                 control.Invoke(new UniversalVoidDelegate(() => controlInvoke
                      (control, function)));
184                 return;
185             }
186             function();
187         }
188 
189         /// <summary>
190         /// Updates the listview to contain orders from the database
              containing station status and notes
191         /// </summary>
192         private void updateList()
193         {
194             FormRestService.ServiceWGetClient rsClient = new
                  FormRestService.ServiceWGetClient();
195 
196             orderConfirmations = new List<OrderConfirmation>();
197             try
198             {
199                 //Clears the list
200                 orderOverViewList.Items.Clear();
```

```
201
202                 //Creates the items in the list
203                 foreach (FormRestService.OrderConfirmation oc in
                      rsClient.getAllActiveOrders())
204                 {
205                     orderConfirmations.Add((OrderConfirmation) oc);
206                     orderOverViewList.Items.Add(new ListViewItem(new[]
207                     {oc.OrderNumber + " " + oc.OrderDate.Day + "-" +
                        oc.OrderDate.Month + "-" + oc.OrderDate.Year,
208                     (oc.StationStatus.Station4 == "Active" ||
                        (oc.StationStatus.Station4 == "Done")? "X": " "),
209                     (oc.StationStatus.Station4 == "Done") ? "X" : " ",
210                     (oc.StationStatus.Station5 == "Active" ||
                        (oc.StationStatus.Station5 == "Done")? "X": " "),
211                     (oc.StationStatus.Station5 == "Done") ? "X" : " ",
212                     (oc.StationStatus.Station6 == "Active" ||
                        (oc.StationStatus.Station6 == "Done")? "X": " "),
213                     (oc.StationStatus.Station6 == "Done") ? "X" : " ",
214                     (oc.StationStatus.Station7 == "Active" ||
                        (oc.StationStatus.Station7 == "Done")? "X": " "),
215                     (oc.StationStatus.Station7 == "Done") ? "X" : " ",
216                     (oc.StationStatus.Station8 == "Active" ||
                        (oc.StationStatus.Station8 == "Done")? "X": " "),
217                     (oc.StationStatus.Station8 == "Done") ? "X" : " ",
218                     oc.Notes.Length + "" }));
219                 }
220                 //Resizes the last column to match the window.
221                 int lastIndex = orderOverViewList.Columns.Count - 1;
222                 orderOverViewList.Columns[lastIndex].AutoResize
                      (ColumnHeaderAutoResizeStyle.HeaderSize);
223             }
224             catch (Exception ex)
225             {
226                 Console.WriteLine("Error: " + ex.Message);
227             }
228         }
229
230         /// <summary>
231         /// Receives an int and return a boolean if there is an update.
232         /// </summary>
233         /// <returns></returns>
234         private bool checkServiceUpdates()
235         {
236             FormRestService.ServiceWGetClient rsClient = new
                  FormRestService.ServiceWGetClient();
237             int serviceUpdates = rsClient.getUpdates();
238             bool result = !(serviceUpdates == updates);
239             updates = serviceUpdates;
240             return result;
241         }
242
243         /// <summary>
244         /// Keep checking for an update every 30 sec
245         /// </summary>
246         private void updateLoop()
247         {
```

```
248             while (true)
249             {
250                 if (checkServiceUpdates())
251                     controlInvoke(orderOverViewList, new Action(updateList));
252                 Thread.Sleep(updateFrequency);
253             }
254         }
255 
256     }
257 }
258 
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10 using WcfService.domain.order;
11
12 namespace SKOffice
13 {
14     public partial class OrderForm : Form
15     {
16         public OrderConfirmation OrderConfirmation { get; private set; }
17         public string OrderName { get; private set; }
18         /// <summary>
19         /// Renames the window with the OrderNumber
20         /// </summary>
21         /// <param name="orderConfirmation"></param>
22
23         public OrderForm(OrderConfirmation orderConfirmation)
24         {
25             this.OrderConfirmation = orderConfirmation;
26             InitializeComponent();
27             Text = "Order - " + orderConfirmation.OrderNumber;
28         }
29
30         /// <summary>
31         /// Makes a status check if there is a link for the Blueprint and
             Requisition button
32         /// </summary>
33         /// <param name="sender"></param>
34         /// <param name="e"></param>
35         private void OrderForm_Load(object sender, EventArgs e)
36         {
37             feedbackBp.BackColor = Color.Red;
38             feedbackReq.BackColor = Color.Red;
39             Console.WriteLine("NOTES: " + OrderConfirmation.Notes.Count);
40             foreach (OrderNote note in OrderConfirmation.Notes)
41             {
42                 addNote(note.Text);
43             }
44         }
45
46         /// <summary>
47         /// Closes the window when you press the button "Close"
48         /// </summary>
49         /// <param name="sender"></param>
50         /// <param name="e"></param>
51         private void closeBtn_Click(object sender, EventArgs e)
52         {
53             Close();
54         }
55
```

```
56          /// <summary>
57          /// Fills the text area with the note text.
58          /// </summary>
59          /// <param name="txt"></param>
60          private void addNote(string txt)
61          {
62              Console.WriteLine("Adding note: " + txt);
63              noteTxt.Text += "-------Start Note------\n";
64              noteTxt.Text += txt + "\n";
65              noteTxt.Text += "---------End Note------\n";
66          }
67      }
68  }
69
```