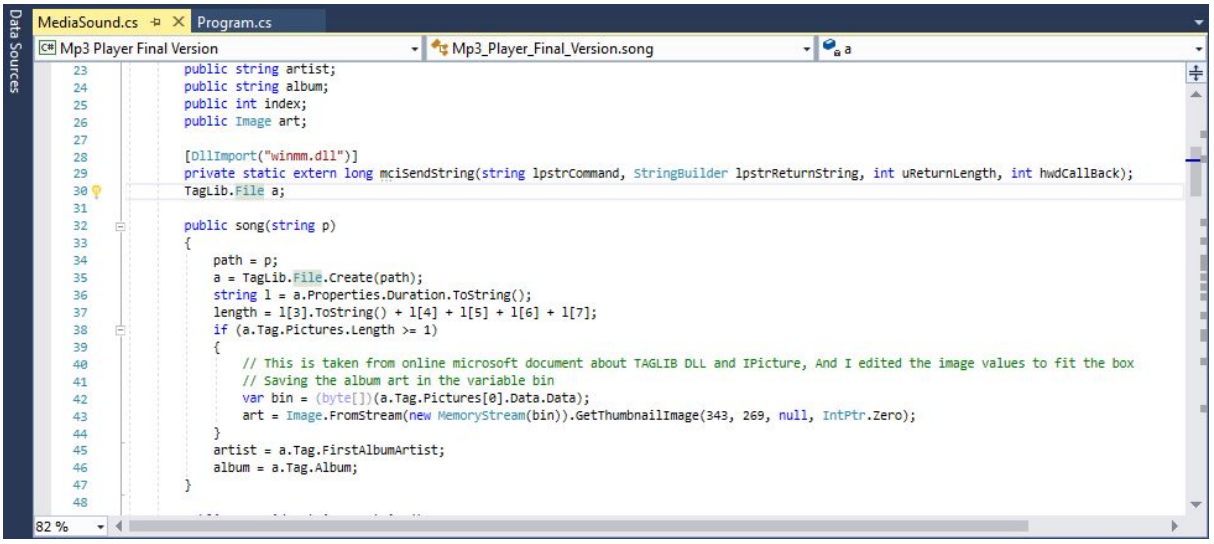
Our project, an mp3 player with very basic functions that can be found in any MP3 player:

- Play, to start the song.
- Pause, to stop the song and pick it off later from the point you stopped in.
- Playlist that can be edited and shuffled to make the best out of it's functioning.
- Sound control by using a trackbar.
- Display the song's information(Title, album, artist, etc…..).

Such a project depended heavily on Object Oriented Programming to create a song class that can carry it's information and details in the form of several attributes while depending on the Double Linked List algorithm to create a fully functioning playlist.



```
MediaSound.cs   Program.cs
Mp3 Player Final Version   Mp3_Player_Final_Version.song   a
23      public string artist;
24      public string album;
25      public int index;
26      public Image art;
27
28      [DllImport("winmm.dll")]
29      private static extern long mciSendString(string lpstrCommand, StringBuilder lpstrReturnString, int uReturnLength, int hwdCallBack);
30      TagLib.File a;
31
32      public song(string p)
33      {
34          path = p;
35          a = TagLib.File.Create(path);
36          string l = a.Properties.Duration.ToString();
37          length = l[3].ToString() + l[4] + l[5] + l[6] + l[7];
38          if (a.Tag.Pictures.Length >= 1)
39          {
40              // This is taken from online microsoft document about TAGLIB DLL and IPicture, And I edited the image values to fit the box
41              // Saving the album art in the variable bin
42              var bin = (byte[])(a.Tag.Pictures[0].Data.Data);
43              art = Image.FromStream(new MemoryStream(bin)).GetThumbnailImage(343, 269, null, IntPtr.Zero);
44          }
45          artist = a.Tag.FirstAlbumArtist;
46          album = a.Tag.Album;
47      }
48
82 %
```

This code snippet explains the process of using tagLib library to extract MP3 details from the file in order to get the artist's name, album title and album art as well. It also shows the constructor being created, thus creating a .txt file that contains the paths of the songs we're importing into our playlist to be streamed. Changing the dimensions of the album art is happening as well to avoid having the picture out of boundaries.

```
Mp3 Player Final Version | Mp3_Player_Final_Version.playlist | start
public void addnext(song s)
{
    if (this.next == null)
    {
        this.next = s;
        this.next.prev = this;
    }
    else this.next.addnext(s);
}

public void play()
{
    delete();
    string Format = @"open ""{0}"" type MPEGVideo alias MediaFile";
    string command = string.Format(Format, path);
    mciSendString(command, null, 0, 0);
    string play = "play MediaFile";
    mciSendString(play, null, 0, 0);
}
public void pause()
{
    string command = "stop MediaFile";
    mciSendString(command, null, 0, 0);
}
public void resume()
{
    string command = " resume MediaFile";
    mciSendString(command, null, 0, 0);
90 %
```

This code snipped displays the process of adding another song into the playlist by taking an object of the data type song we've created, another song will be added to the playlist we created.

The play method will read the path's string and send it to the MCI device in form of a command to play it from the start. The pause button and resume button will also send a command of the same nature to do their corresponding functions as well.

```
134
135         public void addsong(song s, bool write)
136         {
137             if (start == null)
138                 start = s;
139             else
140                 start.addnext(s);
141             s.index = count;
142             count++;
143             if (write == true)
144                 WriteToFile(s.path);
145         }
146
147         void WriteToFile(string p)
148         {
149
150             string[] t = new string[] { p };
151             File.AppendAllLines(path, t);
152         }
153
154         public string[] Read()
155         {
156             string[] paths = File.ReadAllLines(path);
157             return paths;
158         }
159
160         public void clear()
161         {
162             start = null;
163             count = 0;
164         }
165
166         public void Delete(song h)
167         {
168             string[] pth = File.ReadAllLines(path);
169             var y = pth.ToList();
170             y.RemoveAt(h.index);
171             string[] pthto = y.ToArray();
172             System.IO.File.WriteAllLines(path, pthto);
173             count--;
174             h.decindex();
```

For this code snippet, it’s basically an implementation of the double linked list where it adds a song to it and add it’s path to the .txt file we’re using do determine the song’s path.

All the methods are invoked in their respective buttons.

Project was made by:

Emad Maged - Sec 1
Heba Badr - Sec 1
Mazen Zeyad Olama - Sec 3
Mina Maher - Sec 1
Omar Mohamed Sadek - Sec 1
Umar Mahmoud Elbieh - Sec 1
Youssef Mohammed Ali - Sec 1