Digital Jukebox

School of Information Science, Computer and Electrical Engineering
Halmstad University
Box 823, S-301 18 Halmstad, Sweden

June 2011
Acknowledgement:

At first, I want to thank to the Swedloop Company, a company from Halmstad. They provided the wonderful opportunity for me to work with this project, and from this project, I have learned a lot which I have never learned before.

Secondly, I want to thank to my supervisor, Edison Pignaton de Freitas, of Halmstad University, because of his careful supervision and helpful feedback during my thesis. He has given me a lot of useful references and he required me to write a weekly report. He has made good suggestions.

Furthermore, I would also want to say thanks to Kenneth Nillson. At the time I chose my thesis topic, he gave me some suggestions and told me the requirement of the thesis; he gave me a lot of helpful advice.

Last but not the least, I want to thank two friends, Guilherme Gomes and Pauline Droz, who are majoring in marketing and them, too, come from the Halmstad University and they cooperated with me. They suggested the idea and I made an application to realize it. They also gave me some suggestions and gave me necessary help. They contacted the company, Swedloop, and then they gave me the feedback and advice.
<table>
<thead>
<tr>
<th>Details:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors:</td>
<td>Yin Jin</td>
</tr>
<tr>
<td>University:</td>
<td>Halmstad University, Sweden</td>
</tr>
<tr>
<td>Degree Program:</td>
<td>Computer Science and Engineering</td>
</tr>
<tr>
<td>Title of Thesis:</td>
<td>Digital Jukebox</td>
</tr>
<tr>
<td>Supervisor:</td>
<td>Edison Pignaton de Freitas</td>
</tr>
</tbody>
</table>
Abstract

This bachelor’s project uses the Spotify API (Application Programming Interface) to implement a new application called SpotBox, a “digital jukebox”. As it is known, the traditional jukebox has disappeared from the market. There are many reasons for this, such as the limitation of storage capacity and the update frequency of music is not timely. This project proposes to apply the digital technology to build a new jukebox system. The idea is to build an application based on Spotify to run in an ordinary computer that could control the incoming of the coins and the selection of the music. This new version of the jukebox, the digital jukebox, would be deployed in pubs and discos.

The development utilized C# as the programming language and the operation system is Microsoft Windows.

The method of the project has 3 steps. The first step is the application requirements’ analysis. This step identifies and analyses the requirements of the application to work as a digital jukebox and the additional functions based on the Spotify.

The second step is the study about the Spotify API (Application Programming Interface). This is necessary step to verify what is possible to implement from the identified requirements.

Finally, the work was concluded by implementing the required functionalities, as well as enhancing the graphical interface. The graphical interface combined with these functionalities composes an application prototype.

As for the main result, the application was fully developed with the minimum required functionality and interface. Secondary results can be named as the report of the possibilities that allowed by the Spotify API together with the complete requirement analysis of the functionalities that were outside the scope of the intended prototype implementation.
Table of Contents

1. Introduction .................................................................................................................. 1
   1.1 Background ........................................................................................................... 1
   1.2 Objective ............................................................................................................. 2
   1.3 Expected results .................................................................................................. 2
   1.4 Tools .................................................................................................................... 2
   1.5 Outline ................................................................................................................. 3
2. Tools and Methods ...................................................................................................... 4
   2.1 Operating system: Linux/Windows ......................................................................... 4
   2.2 Choosing the programming Languages C/C++/C# ............................................. 4
      2.2.1 Comparison between C/C++ and C# for the development of the project ..... 4
   2.3 Development Tools ............................................................................................. 6
      2.3.1 Microsoft Visual Studio---Microsoft Visual Studio 2010 ........................... 6
      2.3.2 Net Reflector ............................................................................................... 6
      2.3.3 Language C# ............................................................................................... 6
   2.4 Project method ..................................................................................................... 7
3. Requirement Analysis ............................................................................................... 8
   3.1 Use Case Scenario: play the Spotbox ................................................................. 9
   3.2 Non-Functional Requirements ........................................................................... 10
   3.3 Functional Requirements ................................................................................... 11
4. Design and Implementation ..................................................................................... 12
   4.1 Design .................................................................................................................. 12
      4.2 Implementation .................................................................................................. 13
         4.2.1 Log in to the Spotify ................................................................................. 13
         4.2.2 Search a song/artist from the Spotify .................................................... 14
         4.2.3 Play a song from the Spotify .................................................................. 15
         4.2.4 Make a playlist for the digital jukebox ................................................... 18
         4.2.5 Leave a message when playing a song ................................................... 19
         4.2.6 Show the lyrics of a song ................................................................. 20
         4.2.7 Play the top songs in random mode .................................................... 21
      4.3 Graphical User Interface .................................................................................. 22
         4.3.1 Login form ............................................................................................... 23
         4.3.2 The main form of the Digital Jukebox .................................................... 23
         4.3.3 Lyric form ............................................................................................... 25
         4.3.4 Message form ......................................................................................... 26
5. Results ....................................................................................................................... 27
   5.1 The first testing ..................................................................................................... 27
      5.1.1 Added the songs from the local computer ................................................. 27
      5.1.2 Showed the lyrics ....................................................................................... 28
   5.2 The Second testing ............................................................................................. 28
      5.2.1 Search the song from the Spotify and play it ........................................... 28
      5.2.2 Added the song to the playlist ............................................................... 29
5.2.3 Played the top songs randomly .................................................. 29
5.2.4 Leaved a message ...................................................................... 29

6. Conclusion and Future Work .............................................................. 31
  6.1 Conclusion .................................................................................. 31
  6.2 Future work ................................................................................ 32
List of Figures

Figure 2.1 Waterfall model........................................................................................................7
Figure 3.1: Use case diagram modelling the Spotbox functionalities........................................8
Figure 4.1 Module Structure........................................................................................................13
Figure 4.2: The log in form........................................................................................................23
Figure 4.3: Graphical User Interface for the developed prototype........................................24
Figure 4.4: The lyrics form........................................................................................................26
Figure 4.5: The message form....................................................................................................26
Figure 5.1: Add the local song to the playlist............................................................................27
Figure 5.2: Play a song and show the lyrics..............................................................................28
Figure 5.3: Test for searching and adding it to the playlist......................................................29
Figure 5.4: Leave a message......................................................................................................30
Figure 5.5: A message will be shown on the above of the Spotbox..........................................30
List of Listings

Listing 1: Code for logging to the Spotify ................................................................. 14
Listing 2: Code for searching a song/artist from the Spotbox functionality ................. 15
Listing 3: Code for add a song to the playlist and play it from the Spotify .................. 16
Listing 4: Code for the call-back function .................................................................... 16
Listing 5: Code for delivery the music from the Spotify ........................................... 17
Listing 6: Code for handling a song after playing it ................................................... 17
Listing 7: Code for loading the cover of an album ...................................................... 17
Listing 8: Code for adding a song to the playlist ....................................................... 18
Listing 9: Code for the play a song one by one .......................................................... 19
Listing 10: Code for leaving a message ....................................................................... 19
Listing 11: Code for showing the lyrics ...................................................................... 20
Listing 12: Code for load the lyrics ............................................................................ 20
Listing 13: Code for getting the lyrics from the file .................................................. 21
Listing 14: Code for switching the play mode ............................................................ 22
Listing 15: Code for playing in a random mode ......................................................... 22
1. Introduction

This chapter describes the background of the thesis, the goal of the project, expected results, methods and the used software.

1.1 Background

**Jukebox** - A jukebox is a kind of music-playing machine which plays the CDs automatically. The traditional jukebox is a large machine which has a rounded top and it has been decorated in the colorful lights which are around it. The customers can put coins in the machine and choose the songs which they want to hear. After putting the coins into the machine, the customers can use the buttons of letter and number to select a song which they want to hear from a particular album or an artist. It has some disadvantage, such as it needs a lot of space for the CDs or other Medias stored in advance, so it has the limitation of storage capacity and the update frequency of music is not timely. [1]

**Spotify** –Spotify is a kind of music software which the customers can search the song, play the song and manage their own playlist through the internet. It provides a new and fresh way for the customers to enjoy the music through the internet that can offers the users a free and convenient access to a large library of music. Anyone who is willing to use it just needs to register a Spotify account and then download its streaming music player. It is convenient to use Spotify because the users do not need to wait for the music files until they have been downloaded, and the users also do not need to fill up the hard drive before organizing them. [2] [3]
1.2 Objective

The objective of the project is to develop an application called Spotbox that is based on the Spotify API (Application Programming Interface) [4] [5] [6] [7], and it should work as a jukebox, but instead of coins, the expected prototype to be developed will be started by pushing a button. This button will “emulate” the insertion of a coin. Then the user can search the music on line by typing the name of an artist, album or track and hit a button to enter it. Then it will give the results instantly. Finally, it plays the music once.

Spotbox: It is a digital jukebox which is based on the Spotify, but with additional functions and different interface.

1.3 Expected results

The expected result from this project is to have an application meeting with the expectations of the company SwedLoop, which means present a prototype application that can be used to run the ideas of the digital jukebox explained above (SpotBox). The new application does not need to consider the hardware part.

1.4 Tools

To perform the project, the software tools are C# programming over a windows platform and using the Microsoft Visual Studio 2010 software as programming environment. The Spotify API provides the library used in the application development.
1.5 Outline

Chapter 2(Tools and Methods): This chapter analyses which kind of methods that can be used in this project, and the tools, such as which programming language should be used, the supporting operation system and the development software.

Chapter 3(Requirement Analysis): This chapter analysis the study about the application requirements is presented.

Chapter 4(Design and Implementation): This chapter analysis the module structure of the project and the implementation in details.

Chapter 5(Results): This chapter displays the results which are obtained from the test of the SpotBox, including the pictures and instructions.

Chapter 6(Conclusion, and Future Work): This chapter concludes the future work and provides a discussion about more functions that can be added to the SpotBox.
2. Tools and Methods

2.1 Operating system: Linux/Windows

At first, it is important to choose which language to program the application and to choose the operation system for which it will be developed. The Spotify API has been written in C and C#, and this API can be used in both MS Windows and Linux operation systems. Based on the operating primitives and the functionalities offered by the API, the most suitable operating system can be selected. The API documentation presents information showing that both Linux and Windows are suitable for the purposes of the application that is intended to be developed.

2.2 Choosing the programming Languages C/C++/C#

The programming language also has to be chosen. Different languages have their own advantages and disadvantages. The advantages and disadvantages of the language C/C++, and C#, particularly regarding this project, can be listed as follows:

2.2.1 Comparison between C/C++ and C# for the development of the project

C# Advantages:
C# language is a completely new design of the modern language, including the language and its library. It has a strong language ability, because C # language together with. Net has an extent library, so C # is appropriate to deal with the development of the applications. The programmer can directly call some methods with entire functionalities that diminish the development effort. So it is easy to develop. The library is also easy to study and get familiar with it. It is high-level language, and through the mechanism of the underlying virtual machine, common
errors in C / C++ language, such as memory leaks and security issues can be avoided. [8][9]

**C# Disadvantage:**
1. It is not suitable to develop for the low-level functions.
2. The running efficiency is not better than the C/C++ language.
3. Before client use C# to develop it, client should install the .net framework. [8][9]

**C/C++ Advantage:**
C/C++ is lower-level language, compared to C#, so they have provided more flexibility to access function primitives and customized code. They allow developers to freely dispose of the memory. They are suitable for low-level development, for example: device drivers, services, network communications, and even directly used to write the operating system. C/C++ does not need the support of any framework, such as .Net. They can generate the binary file that can be run directly. [10][11]

**C/C++ Disadvantage**
Using C/C++ requires attention in the use of memory and it is easy to make mistakes in relation to this usage. It requires more development effort to provide functionalities for the target of application, which are somehow basic in other languages that use for supporting frameworks, such as Java and C#. [10] [11]

**Conclusion:**
Compared the C/C++ with C#, choosing which language to use in the development of a project depends on the specific situation and the project needs. If it is required to have the high efficient and have the possibility to use low-level system calls, choosing the C/C++ is the best option. Otherwise, C# provides an easier platform with richer library that shorten the development time.
C language is procedure oriented language, which has good portability, just requiring the compilation of the code for a given target system. C++/C# language both are procedure object language. They both have good portability, but compared to procedural languages, they provide better maintainability.

So from the comparison between these options, the choice for C# language to develop the application is based on the fact that it provides a richer library and it can reduce the time for the application development, as the low-level resource access provided by C/C++ is not needed.[8][9][10] [11]

### 2.3 Development Tools

#### 2.3.1 Microsoft Visual Studio---Microsoft Visual Studio 2010

The Microsoft Visual Studio is a kind of powerful programming software which guarantees the whole application of life cycle within the quality code during the time of the design to the disposition. If the users want to develop for the SharePoint, a web page, Windows, Windows Mobile, or the subsequent application. This outstanding programming software can provide the users all in-one solution. [12]

#### 2.3.2 Net Reflector

Net Reflector is powerful software which can help the user to understand the code library. The user can search, analyze, debug and decompile .net component and code. [13]

#### 2.3.3 Language C#

As C# has been analyzed before, it is more suitable for the goals of this thesis. It is more convenient to use C# to develop an application, due to the reasons presented above.
2.4 Project method

The method of this project uses a waterfall development approach. As the figure 2.1 shows [14], the Waterfall model is a kind of approach which can be used for the project development process. [15] Every step has its feedback information, it will not go back unless one step has found some problems, it will go back to the previous step and make some appropriate changes until it has been completed, and then the software development process will flow from the current step to the next step. The Waterfall model divides the software life cycle into 6 sequential steps: 1. Feasibility study 2. Requirements analysis 3. Software design 4. Software coding and testing 5. Integration and system testing 6. Maintenance. [14] [15]

This section describes the steps 2 (Requirement analysis), 3 (Software design) and 4 (Software coding and testing). Step 1 was already described above, by analyzing the feasibility of the project and selecting the appropriate tools. Steps 5 (Integration and system testing) and 6 (Maintenance) are out of the scope of the thesis.

![Figure 2.1 Waterfall model [14]](image-url)
3. Requirement Analysis

The goal of the requirement analysis [14] [16] is to get all the relevant information about the product which will be developed, understanding what kind of requirements the customer really needs.

Use case diagram [17] is a kind of diagram which is used for observing the roles of actors in a system with several functional perspectives. It can show a number of actors and a number of use cases, which can represents the project functionalities, dependencies and the relationships between them. Use cases [18] are a kind of software model which is used to describe the requirements of the project, and then it can help the programmers to make the understanding of the project functionalities more clearly and it can also help them to determine which functions need to be implemented.

In order to describe the functional requirements more clearly, Use cases diagram for the Spotbox is listed as follows:

Figure 3.1 shows the use case diagram for the Spotbox, in which there are two actors. One is the User, and the other one is the Administrator.

![Use case diagram modelling the Spotbox functionalities.](image-url)
The following functionalities are identified for the system and are classified according the actor that performs it:

**For User:**

1. The user can search the song which the user wants to play.
2. The user can add the song to the playlist.
3. The user plays the song which the user just added to the playlist.
4. The user can show the lyrics of the song.
5. The user writes a message, and then wants to show it on the above of the application.
6. If no one selects a song to play, the user can play the top songs in a random mode.
7. The user can add the song from the local computer.

**For Administrator:**

1. The administrator should log in the Digital Jukebox.
2. The administrator can add the songs which are in the local computer.
3. If no one selects a song, the administrator plays the top songs in a random mode.
4. The administrator can log out the Digital Jukebox.

**3.1 Use Case Scenario: play the Spotbox**

**For the users:**
For the administrator:

Use Case Scenario: play the Spotbox
1. The Administrator enters the username and the password to log in the Spotify.
2. If there is no song to play, the Administrator can press the button, and then the digital box will play the top songs in a random mode.
3. The Administrator can add some songs from the local computer which is not available in the Spotify.
4. The Administrator log out the Spotify.

3.2 Non-Functional Requirements

Graphic User Interface: A friendly graphic user interface should be offered by the digital jukebox to the user. The graphic user interface should be clear and easy for the customer to use it. The Graphic User Interface is not supplied by the Spotify API.
3.3 Functional Requirements

The function can be listed as follows:

- Develop a playlist system in which customer is allowed to program only one song per time without interrupting the current one. The new song programmed will always be the last one on the playlist.
- After having programmed a song, the software should show a box in which customer can write his/her name, a message among other information, if the user wants and show it in the playlist with the title of the song.
- Karaoke functions with lyrics which appear on the screen.
- Charts list: Top ten of the machine. To develop a system by which the Spotbox machine starts automatically to play the songs of the top 40.
- Create an interconnection between Facebook through which the message will automatically posted on the Facebook of the customers if they do a log in.
4. Design and Implementation

4.1 Design

This chapter explains the design and implementation the functions to make a complete digital jukebox at first, the functionalities came with the prototype, and then the graphic user interface.

The goal of the design is to transform the requirements of the project into a program structure which is the appropriate to be implemented in the next step. The project design is use the traditional design approach. It consists of two different steps: Structure analysis and Structure design. [14] [16]

The first step is to implement all the functions step by step, and then design the graphic user interface.

The application decomposed into 2 main modules: 1. Login module 2. Mainbox

As figure 4.1 shows, the Mainbox decomposed into several sub-modules: 1. Play the local song 2. Play the song from the Spotify 3. Lyrics (Karaoke) 4. Leave a message 5. Link to the Facebook. The module “Play the local song” decomposed into 2 sub-modules: 1. Add song 2. Play song. The module “Play the song from the Spotify” has 3 sub-modules: 1. Search song 2. Add song to the playlist 3. Play song

The module “Play a song” has 3 sub-modules: 1. Play the song one by one 2. Play the top songs in random mode 3. Browse the cover of the album

The module “Play song”, “Play the song one by one” and “Play the top songs in random mode” use the same class to play the song through the Spotify.

The module “Add song” and “Add song to the playlist” use the same class to save the playlist. Others modules are dependent.
4.2 Implementation

The objective of the software implementation is to transform the software design (the third step of the Waterfall model) into the program source codes. Every step of the design will be implemented to be a software module. Each module will be tested individually, because not only it can provide the most efficient manner to debug the programming error, but also the other module which may not be ready has to be interfaced. [14] [16]

4.2.1 Log in to the Spotify

This functionality is coded by handlers of these buttons (button2_Click method). After pressing the button, if the administrator types the right username and the password, it will call the *hide* function (Line 4 of the Listing 1), then the “log in” form will hide, the object *s* of the Session (Line 5 of the Listing 1), it will call the *GetSpotifySession* function which is in the Program.cs. It will create a new object *s* of
Session in the Mailbox (Line 6 of the Listing 1) and give the value to the `m_mainbox` which is declared in the Program.cs (Line 7 of the Listing 1). Then call the `show` function, and then the `mainbox` form will appear.(Line 8 of the Listing 1)

```
1. private void button2_Click(object sender, EventArgs e)
2. {
3.   ...
4.   this.Hide();
5.   Session s = Program.GetSpotifySession();
6.   MainBox mainBox = new MainBox(s);
7.   Program.m_mainbox = mainBox;
```

Listing 1: Code for logging to the Spotify

### 4.2.2 Search a song/artist from the Spotify

The application presents several buttons in its GUI which trigger the executions of its functionalities. This functionality is coded by handlers of the button (button2_Click method).

After pressing the button that starts the application, it searches the song or the artist which the user typed in the textbox. The button2_Click function implements the stating function (line of 1 in the Listing 2). The search result is the style of `AlbumSearchResult` (line of 5 in the Listing 2). It calls the `SearchManager` function (line of 5 in the Listing 2) to get the searching result. The local variable `Success` (line of 6 in the Listing 2) is a `Boolean` used by the class `AlbumSearchResult` to check whether it will continue running. The local variable `Success` (Lines 6 of the Listing 2) is turned to be true, in this moment it will call the `foreach` function to make the results in an order traversal. Then it will find the name of the songs and the artist which will convert into strings (Line 13,14 of the Listing 2), they will be added to the `listbox1` (Line 15 of the Listing 2).
1. Private void button2_Click (object sender, EventArgs e)
2. listBox1.Items.Clear();
3. try
4. {
5.   AlbumSearchResult result =
6.   SearchManager.Instance.SearchAlbum (textBox1.Text, 1);
7.   if(result.Success)
8.     {
9.       ... 
10. foreach(Spotikat.Spotify.Entity.Album album in
11.   result. Albums)
12.     {
13.       if (album.Availability.Contains("worldwide") ||
15.         {
16.             String name = album.Name.ToString();
17.             String artist = album.Artist.Name.ToString();
18.             listBox1.Items.Add(name + "--" + artist);
19.         }
20.     }
21.   catch (SpotifyCommunicationException scex)
22.     {
23.       listBox1.Items.Add("Failed to communicate with
24.       Spotify");
25.   }

Listing 2: Code for searching a song/artist from the Spotbox functionality

4.2.3 Play a song from the Spotify

Playing a song from the Spotify is through the stream. Button9_Click function (Line 1 of the Listing 3), the application will start to play a song from the Spotify. The global variable isPreview(Line 3 of the Listing 3) has been declared before, its initial value equals false. It will control how to play the songs. The user can play a song directly or add it to the playlist first. The variable s (Line 4 of the Listing 3) is the selected song which the user pressed. Search function (Line 7 of the Listing 3) will
search the songs through Spotify. And then it will call the call-back function (Line 3, 4, 5, 6 of the Listing 4) one by one. Listing 4 is the music delivery call-back function. The local variable e.Samples (Line 1 of the Listing 5), if length bigger then 0, creating a new constructor function bassplayer (Line 7 of the Listing 5). Listing 5 is after finishing playing a track, and then will call this call-back function. Because playing a song has two ways, one is play it directly after searched it, the other is playing it after add it to the playlist, so it needs a variable to control the 2 situation. The variable isPreview (Line 4 of the Listing 6) has been declared before, if it is false, just play this song, if it is true, after playing this song, it will continue playing the songs one by one or randomly. Listing 7 is loading a cover of an album call-back function. When the time the song start playing, if it has a cover of an album (Line 4 of the Listing 7), and then it will load this cover and save it as a format of JPEG (Line 8 of the Listing 7).

Listing 3: Code for add a song to the playlist and play it from the Spotify

```csharp
1. private void button9_Click(object sender, EventArgs e)  
2. {  
3.     Program.isPreview = true;  
4.     string s = listBox1.SelectedItem.ToString();  
5.     char sw = '-';  
6.     string[] ret = s.Split(sw);  
7.     session.Search(ret[0], 0, 500, 0, 500, 0, 500, null);  
8.     textBox1.Text = ret[0];  
9. }
```

Listing 4: Code for the call-back function

```csharp
1. public static Session GetSpotifySession()  
2.     ...  
3.     s.OnSearchComplete += HandleOnSearchComplete;  
4.     s.OnMusicDelivery += HandleOnMusicDelivery;  
5.     s.OnEndOfTrack += HandleOnEndOfTrack;  
6.     s.OnImageLoaded += HandleOnImageLoaded;
```
1. static void HandleOnMusicDelivery(Session sender, MusicDeliveryEventArgs e) {
    if (e.Samples.Length > 0) {
        if (player == null) {
            player = new BassPlayer();
            ...
        }
        e.ConsumedFrames = player.EnqueueSamples(new AudioData(e.Channels, e.Rate, e.Samples, e.Frames));
    }
}

**Listing 5: Code for delivery the music from the Spotify**

1. static void HandleOnEndOfTrack(Session sender, SessionEventArgs e) {
    if (isPreview) {
        ...
    } else {
        playNext = true;
    }
}

**Listing 6: Code for handling a song after playing it**

1. static void HandleOnImageLoaded(Session sender, ImageEventArgs e) {
    if (e.Image != null) {
        try {
            ...
        }
    }
}

**Listing 7: Code for loading the cover of an album**
4.2.4 Make a playlist for the digital jukebox

After the users has searched a song, and then can add it to the playlist if don not want to play the song directly. In the button5_Click function, the string s (Line 3 of the Listing 8) is the item which the users selected in the search result. The song result will include the artist. The char sw(Line 4 of the Listing 8) is a ‘-’, and then call the function Split(Line 5 of the Listing 8) will remove the content which is after the ‘-’, such as the name of the artist. The selected item will be added to playlist (Line 6 of the Listing 8). If the global variable playnext is true, and if the index of the selected (Line 6 of the Listing 9) is smaller than the total number of the song(Line 7 of the Listing 9)minus one, the index of the selected will plus one(Line 11 of the Listing 9)which means the Digital Jukebox will play the next song, otherwise the index will equal to 0(Line 15 of the Listing 9)which means the Digital Jukebox will play the next song from the first song in the playlist.

Listing 8: Code for adding a song to the playlist

```
1. private void button5_Click(object sender, EventArgs e) { 
2.     string s = listBox1.SelectedItem.ToString(); 
3.     char sw = '-'; 
4.     string[] ret = s.Split(sw); 
5.     listBox3.Items.Add(ret[0]); 
6.     ... 
```
1. private void timer2_Tick(object sender, EventArgs e)  
2. {  
3.    ...  
4.    if (Program.playNext == true)  
5.    {  
6.        int i = listBox3.SelectedIndex;  
7.        int num = listBox3.Items.Count;  
8.        ...  
9.        if (i < num - 1)  
10.           {  
11.               listBox3.SelectedIndex++;  
12.               ...  
13.           }  
14.           else  
15.           {  
16.               listBox3.SelectedIndex = 0;  
17.               ...  

Listing 9: Code for the play a song one by one

4.2.5 Leave a message when playing a song

When a song is starting to play, the button1_Click function, it will display a new window for the user to type the words. The content in the new window will be copied to the label on the above of the application. (Line 3 of the Listing 10)

1. private void button1_Click(object sender, System.EventArgs e)  
2. {  
3.    mF_Form.label1.Text = richTextBox1.Text;  
4.    ...  

Listing 10: Code for leaving a message
4.2.6 Show the lyrics of a song

If the user wants to show the lyrics, call the radioButton1_CheckedChanged function then the lyrics will be displayed on the above of the Spotbox.

But the lyrics of a song is not available through the Spotify, it should be download in advance. The radioButton1_CheckedChanged function, it loads the lyrics (Line 4 of the Listing 11), it will call the funcation Load_Lyrics(Line 1 of the Listing 12), in the Load_Lyrics function, it will call the funcation Read_Lyrics(Line 3 of the Listing 12).

The variable read (Line 7 of the Listing 13) will create a string buffer to store the lyrics one line for once time and it can store the string stream continuously. The clear () function will clear the one sentence of the lyrics after it has been showed out. It will have a judgement (Line 9 of the Listing 13), it will add until the lyrics equals null. Adding the one line lyric for once time and type “enter” (Line 11 of the Listing 13), it will add up the lyrics (Line 12 of the Listing 13).

Listing 11: Code for showing the lyrics

```java
1. private void radioButton1_CheckedChanged(object sender, EventArgs e)
2. {
3.     ...
4.     oFormLrc.Load_Lyrics(@"I need a doctor.lrc");
5.     oFormLrc.Show();
6.     ...
```

Listing 12: Code for load the lyrics

```java
1. public void Load_Lyrics(string File)
2. {
3.     Read_Lyrics(File);
4.     ...
```
1. void Read_Lyrics(string file)
2. {
3.     ...
4.     try
5.         {
6.             ...
7.                 read = new StreamReader(stream, Encoding.GetEncoding();
8.                     lyricslist.Clear();
9.                 while (!read.EndOfStream)
10.                     {
11.                         line = read.ReadLine() + "\r\n";
12.                         totallines += line;
13.                         match_lyrics(line);

Listing 13: Code for getting the lyrics from the file

4.2.7 Play the top songs in random mode

If there is no one selects a song, the button6_Click function will switch the playing mode, and then the Digital Jukebox can play the top songs in a random mode. The global variable isRandom is to control the play mode (Line 3 of the Listing 14). The global variable playnext(Line 3 of the Listing 15) which has been declared in the program.cs before, it can control the Digital box whether it needs to play the next song. The global variable isRandom(Line 5 of the Listing 15) which also has been declared in the program.cs before, it is control the Digital box to play the song randomly or play it one by one. Creating a constructor function Random (Line 9 of the Listing 15) and ro (Line 9 of the Listing 15) is a seed number. The random number in the range of the playlist will created by the function ro.next(Line 9 of the Listing 15). This value will be given to the variable iResult(Line 10 of the Listing 15). Then it will search this song and play it(Line 14 of the Listing 15), and then set the global playNext to false.(Line 15 of the Listing 15)
1. private void button6_Click(object sender, EventArgs e)  
2. {  
3.   Program.isRandom = !Program.isRandom;

Listing 14: Code for switching the play mode

1. private void timer2_Tick(object sender, EventArgs e)  
2. ...  
3. if (Program.playNext == true)  
4. {  
5.   if (Program.isRandom == true)  
6.   {  
7.     int iResult;  
8.     ...  
9.     Random ro = new Random();  
10.    iResult = ro.Next(0, num);  
11.   ...  
12.    listBox3.SelectedIndex = iResult;  
13.   ...  
14.    session.Search(s, 0, 500, 0, 500, 0, 500, null);  
15.    Program.playNext = false;  
16.   ... 

Listing 15: Code for playing in a random mode

4.3 Graphical User Interface

As it has been mentioned before, the Spotify API does not supply the GUI. Spotbox is a kind of music application and it should have a friendly GUI for the customer to use it.

In the visual studio 2010, first is to create a new C# windows application form, then add some components based on its application form. There are 4 forms, but the main
form is the musicbox form. Majority of the function will be display on it. Other 3 forms are login form, lyrics form and the message form.

4.3.1 Login form

As shown in figure 4.2, before using the Digital Jukebox, the administrator should enter the username and the password to log in. The username the form has been remembered, but if the administrator enters the wrong password, it cannot log in until the password is correct.

![Figure 4.2: The log in form](image)

4.3.2 The main form of the Digital Jukebox

This is the main form, after the administrator has logged in and this form will be displayed as the fellow figure3.2. Then the user can start to use it.
Figure 4.3: Graphical User Interface for the developed prototype

From the above main form, there are some components, every components has their own function.

Textbox Component:
Textbox1: This textbox is for the customer to search the song whether it exist in the Spotify or not.
Textbox3: This textbox will record the time when the form is showing the lyrics of the songs.
Textbox4: This textbox will record the total time when the form is showing the lyrics of the songs.

Button Component:
Button1 (leave an msg): When the digital jukebox is playing song, the customer can leave a message, and then this message will be showed on the above of the digital box.
Button2 (search): This button is used after typed the name of the song or the singer in textbox1, then press this button, it will search the results automatically.
Button3 (add song): This button is for the administrator to add the music which can’t find in the Spotify, just press it and add the music which is in the local computer.
Button 4 (facebook): Press this button; it will be linked to the Facebook web site directly.

Button 6 (switch): Press this button; it will switch between the play mode: play the songs one by one or play the top songs in random mode.

Button 9 (preview): Press this button, the song which you selected will be played soon.

**RadioButton Component:**

RadioButton(ShowLyrics): If the user press it, it will show the lyrics of the song.

RadioButton(Stop): If the user press it, it will stop showing the lyrics of the song.

RadioButton(Close): If the user press it, it will close the lyrics of the song.

**Listbox Component:**

Listbox 1: After the customer searches the song or the singer or the name of the album, all the results will be showed in the listbox 1.

Listbox 2: After the administrator adds the songs from the local computer, all the songs from the local computer will be showed in the listbox 2.

Listbox 3: After the user searches a song which he likes, then the user can add it to the listbox 3, it is a playlist.

Listbox 5: This is the cover of the album, if the customer plays a song which from the Spotify stream, then it will change to the cover of its album.

**PictureBox Component:**

PictureBox 3: It will update the cover of the album when it starts playing a song.

### 4.3.3 Lyric form

This form will display the karaoke functions with lyrics which appear on the screen.
4.3.4 Message form

After the user has chose a song and played it, if the user wants to leave a message, and then after pressing the button “leave a msg”. After that the message form will be appear, the user can type the words which the user wants to show on the screen. As shown in figure 4.5

Figure 4.4: The lyrics form

Figure 4.5: The message form
5. Results

This chapter covers the functions of the Digital Jukebox which has been designed and all its functions will be explained and it also shows a result summary which is got by the test.

The Digital Jukebox was tested on the MS Windows and the development environment is Visual Studio 2010.

Testing was divided into two parts. One part was added and played the songs from the local computer, showed the lyrics and show the top list of the local songs, the other part was searched a song and played it, added the song to the playlist, played the top5 songs randomly and left a message when playing a song.

5.1 The first testing

5.1.1 Added the songs from the local computer

If a song is not available in Spotify, the user can try pressing the button” As Figure5.1showed, the user just pressed the button ”Add the song”, then it existed a windows, it would supply to the user for choosing the song which the user wanted to add to the local playlist.

![Figure 5.1: Add the local song to the playlist](image)
5.1.2 Showed the lyrics

As the Figure 5.2 shows, after pressing the radio button “ShowLyrics”, then the lyrics would display on the top of the application.

![Figure 5.2: Play a song and show the lyrics](image)

5.2 The Second testing

5.2.1 Search the song from the Spotify and play it

As the Figure 5.3 shows, the user typed the name of the song/artist in the blank, and then pressed the button “search”, the result which was associated with the content the user typed would be listed in the table. Then pressed the button “preview”, the song selected by the user would be played.
5.2.2 Added the song to the playlist

As the Figure 5.3 shows, after that the user chose a song, and then pressed the button” add”, the song selected by the user would be added to the playlist.

5.2.3 Played the top songs randomly

As the Figure 5.3 shows, if nobody selected a song, the user/administrator can press the button “switch”, and then the Spotbox would play the top songs automatically in a random mode.

![Figure 5.3: Test for searching and adding it to the playlist](image)

5.2.4 Left a message

As figure 5.4, when a song was playing, the user wanted to leave a message, and then just pressed the button” leave a msg”, it would displayed a new form which the user
can typed the words. For example: The user pressed the button, and then it displayed a new form. After that, the user typed “Hello everyone”

Figure 5.4: leave a message

As the Figure 5.5 shows, then the message would showed on the above of the Application:

Figure 5.5: A message will be shown on the above of the Spotbox
6. Conclusion and Future Work

6.1 Conclusion

This project was required to make an application which can works as a digital jukebox. Almost the functions which were related to Spotbox had been finished: Searching a song, adding it to the playlist, and it can play the song one by one. The user can leave a message when the song is playing, playing the top songs in a random mode and display the lyrics on the screen (karaoke function). During the time of testing, some songs which were not available in the Spotify, so the Spotbox had been added some additional functions, such as adds the song from the local computer and play them. However some functions are not totally completed and they need enhancements, such as: Show the lyrics on the screen and create an interconnection with Facebook and paste a message automatically on it.

The user can search the song from the Spotify, and then the user can play the song directly or add the song to the playlist if somebody is playing the song now. The song will always add to the playlist in the last place which is in a FIFO style (first in first out). During the time when a song is being playing, the user can leave a message, and then type the words which the user wants to show on the above of the Spotbox. There is button to control the Spotbox, if the user or the administrator wants to play the top songs in a random mode, and then just presses the button, and then the Spotbox will start to play the top songs randomly. If the user searches the song which is not available in the Spotify, the user can add it from the local computer. This is an additional created function.

However there are 2 functions needs to be improved as mentioned before. One is showing the lyrics, the other is the interconnection with the Facebook.
6.2 Future work

Regarding the requirement of this thesis, the Spotbox is almost completed successfully. However it also has several parts need to be improved and still have much future work that can be done.

The first is about the problem of the thread. This application is using Single-Thread, which is enough to fulfill the current requirements, so it is not necessary to use the Multi-Thread. In the future, someone will add more functions based on this project, may be the use of Single-Thread will not enough to fulfill the requirements, so the developers will improve it with using the Multi-Thread because it is more efficient.

About the lyrics, because there is not a big Database which can supply all the lyrics of the songs on the Internet now. The Spotbox has to download the lyrics in advance and shows it on the screen. In the future, it is necessary to find a website which can supply all the lyrics of the songs from the Spotify or create a database which will contain all the lyrics of the songs.

Spotify supplies plenty of songs through the Internet. However some famous or the classical songs cannot be found in the Spotify. Adding an additional function is necessary: the Digital Jukebox can add the local songs to play. In the future, if it is necessary and possible, using the API from other software like Spotify to develop which can supply some old and classical songs or downloads as much as the songs in advance which are not available in the Spotify.

About the Facebook, now the Spotbox cannot paste the message automatically on the Facebook, so wait the API of the Facebook come out in the future.

The Spotbox can also develop an information system (Bluetooth or “FILE”) in the Spotbox machine which can communicate with cell phones of the customers who has the Spotbox software in its mobile to create an exchange of information. For example: (1) the customers can program their song from their mobile; (2) the Spotbox mobile software has already the log in of the Twitter and Facebook; therefore it is automatically linked with these websites.
Reference

[1]. The jukebox Website: http://www.tititudorancea.com/z/jukebox.htm
[2]. The Spotify Website: http://www.spotify.com/se/about/what/
[3]. The Spotify Website: http://www.macupdate.com/app/mac/33033/spotify
[4]. The Spotify API Website: http://developer.spotify.com/en/libspotify/overview/
[5]. The Spotify API Website: http://getsatisfaction.com/spotify/topics/announcing_a_c_net_wrapper_for_libspotify
[6]. The Spotify API Website: http://libspotify-sharp.hallerud.se/
[7]. The Spotify API Website: http://github.com/jonasl/libspotify-sharp.
[12]. The Microsoft Visual Studio Website:
    http://www.microsoft.com/visualstudio/en-us/home
[13]. The .Net Reflector Website: http://www.reflector.net/
[15].The Waterfall Model Website: http://www.idinews.com/waterfall.html