一、项目分工

学号	名字	角色	班级	职责	贡献
14331210	莫浩然	组长	周二班	负责 MVVC 模式构建，newpage 主页面的设计，歌曲的各项操作功能	40%
14331211	莫华晓	组员	周二班	负责 mainpage 登陆界面的设计和用户设置 settingpage 页面的设计	40%
14332006	谷溢	组员	周四班	负责 newpage 主页面歌词、歌曲图片的网络访问功能	20%

二、开发环境

1) 电脑系统：Win10
2) 编程 IDE：VS 2015 Community
3) 编程语言：C# 与 XAML

三、项目阐述

1. 名称

CoolDog 音乐播放器

2. 简介

CoolDog 音乐播放器是一款参照酷狗音乐而设计的为音乐爱好者专门打造的 Win10 应用。用户注册账号后可以登录进入主界面，然后可以在本地选取喜欢的歌曲（.mp3 格式），添加到播放列表，然后进行播放、暂停、停止、上下首播放等功能。同时在主界面支持退出登录，或者跳转到用户信息修改页面，修改用户信息比如用户密码等。

3. 功能

1) Adaptive UI

主页面（NewPage）UI 有宽屏和窄屏两种。宽屏时，左边显示歌曲播放列表，以及歌曲的各种操作按钮；右边则显示播放歌曲的歌词，以及歌手的图片。窄屏时，只显示左边的歌曲播放列表，以及歌曲的各种操作按钮。

2) Data Binding

对歌曲使用了 Models、Viewmodels 的模式，在主页面歌曲播放列表那里，对于每一首添加的歌曲，绑定其歌曲名、歌手名然后显示出来。

3) Database

设置了 2 个数据库，一个是针对用户的 Users 数据库，一个是针对歌曲的 Songs 数据库。在登陆/注册页面（MainPage），注册了用户后，便把用户信息插入到 Users 数据库。如果是需要登陆，则在数据库查询是否有相关用户信息，若有，则登陆成功，进入主页面（NewPage）。

主页面有修改用户信息的跳转功能，跳转到信息修改页面（SettingPage），可以进行信息的修改，修改成功，即更新 Users 数据库的相关内容。

进入主页面（NewPage），会遍历 Songs 数据库，查找到属于该用户的歌曲（两个数据库有相同的键 username 和 user，两个相同即说明是属于该用户的歌曲），并加到播放列表。同时该用户添加歌曲时候，也会把歌曲的相关信息插入到 Songs 数据库。同时也可以进行歌曲信息在 Songs 数据库里的删除功能。

同时，主界面还有一个查询歌曲信息的区域。在文本框里面输入歌曲名或者歌手名，就能够通过查询数据库，找到含有该信息的记录，然后显示出来。

4) App to app communication

在主页面（NewPage），歌曲有 share 的菜单按钮，点击后可以分享歌曲的信息到邮件以及 OneNote，实现 APP 之间的通信功能。

5) Network accessing

通过网络访问，获取歌曲的歌词以及专辑封面图片。

6) File management

在主页面（NewPage），能够进行文件的读取。这里主要利用 openPicker 类实现打开文件夹选取.mp3 文件，然后存储到 StorageFile 类里面，再依次读取文件名。

7) Live tiles

主页面（NewPage）也同样实现了自定义磁贴的功能，点击 update tiles 按钮后，磁贴会根据不同的大小显示不同内容。

8) 自己实现功能——利用 MediaElement 类实现歌曲的各种功能：

首先在 NewPage.xaml 里面插入了一个 MediaElement 控件，然后点击歌曲列表中某首歌时候，将歌曲的绝对路径加入到 MediaElement 控件的 Source 里面，然后通过 MediaElement.Play()函数即可实现选中歌曲的播放功能。同时，当调用 MediaElement.Pause()函数，或者 MediaElement.Stop()函数时候，能实现对选中歌曲的暂停、停止功能。而要实现上/下首功能，则需要每次点击上/下首按钮时候，改变 MediaElement 的 Source，再调用 MediaElement.Play()等函数。

当然，上面说的只是最简单是实现歌曲播放、暂停、停止功能的基本思路。但是实际操作的时候会发现 MediaElement 类的 Play()、Pause()、Stop()功能会根据 MediaElement.CurrentState 的不同而有不同的效果（会在下面项目难点部分详细说明），所以最后是在主页面设置了 3 个 MediaElement 类，来针对上一首、现在播放、下一首歌的功能实现。

4. 亮点

1) 实现了 project 要求的所有知识点外，还增加了音频播放的相关功能：从本地添加歌曲、播放、暂停、停止、上下首。
2) 实现了两个数据库的交互，即能查询两个数据库中主键相同的记录：登陆的用户名以及歌曲数据库中该用户名对应的歌曲，实现不同用户保存不同的歌曲。同时实现用户信息的设置以及保存到数据库中。
3) 播放歌曲的同时，可以通过网络访问，获取歌曲的歌词以及专辑封面图片。

5. 实验步骤&功能对应代码解释

1) 创建 user 数据库表，里面分别保存用户的 username，password，注册 datetime，注册成功后设置的 nickname，email，phone，qq，wechat，sex 性别：

```
conn = new SQLiteConnection("user.db");
string sql = @"CREATE TABLE IF NOT EXISTS
             Users (id            INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
                                  username       VARCHAR(140),
                                  password       VARCHAR(140),
                                  datetime       VARCHAR(140),
                                  nickname       VARCHAR(140),
                                  email          VARCHAR(140),
                                  phone          VARCHAR(140),
                                  qq             VARCHAR(140),
                                  wechat         VARCHAR(140),
                                  sex            VARCHAR(140)
                                  );";
using (var statement = conn.Prepare(sql))
{
    statement.Step();
}
```

2) 设计主界面 mainpage 的登陆页面的 UI

3) 在 mainpage 实现 user_login 和 user_register 功能：用户登录的功能主要是通过从数据库中读取用户姓名，如果查找到匹配的用户名字，比较输入的密码是否与注册的密码一致，如果一致的话则跳转到 newpage 页面，否则弹出错误信息提示框：

```
private void user_login(object sender, RoutedEventArgs e)
{
    using (var statement = App.conn.Prepare("SELECT username, password FROM Users
        WHERE username LIKE ? AND password LIKE ?"))
    {
        bool judge = false;
        statement.Bind(1, username.Text);
        statement.Bind(2, password.Password);
        while (statement.Step() != SQLitePCL.SQLiteResult.DONE)
        {
            judge = true;
            string user = username.Text;
            Frame.Navigate(typeof(NewPage), user);
        }
        if (judge == true) return;
        else
```

```
            var i = new MessageDialog("Login fail! ").ShowAsync();
    }
}
```

而对于用户注册功能，首先判断数据库中是否存在名字相同的用户，如果存在相同用户则注册失败，弹出错误提示框：该用户已经成功注册。当用户成功注册后将用户信息写入到数据库中，实现注册成功功能（由于代码类似则不放出）。

途中还实现了一项功能就是鼠标点击 textbox 时用户名和密码置为空，毕竟感觉一个个删除用户名和密码有点反人类：

```
private void Mouse_In(object sender, RoutedEventArgs e)
{
    username.Text = "";
    password.Password = "";
}
```

4) 同时，还实现了修改用户信息的功能，在 newpage 页面中可以通过 flyoutitem 的 setting 按钮跳转到 settingpage 的页面修改用户信息。

5) 在 settingpage.xaml.cs 中主要实现两个函数的功能，一个是 settingbtn 主要是保存用户的个人信息到数据库中，而 passwordbtn 则支持用户更改自己的密码：

```
using (var statement = App.conn.Prepare("UPDATE Users SET nickname = ?, email = ?,
    phone = ?, qq = ?, wechat = ?, sex = ? WHERE username = ? "))
{
    statement.Bind(1, nickname.Text);
    statement.Bind(2, email.Text);
    statement.Bind(3, phone.Text);
    statement.Bind(4, qq.Text);
    statement.Bind(5, wechat.Text);
    if (male.IsChecked == true)
        statement.Bind(6, "male");
    else
        statement.Bind(6, "female");

    statement.Bind(7, user);
    statement.Step();
}
Frame.Navigate(typeof(NewPage), user);
```

而类似的passwordbtn也是如此，通过对username的确定，修改用户的密码，修改成功后返回到newpage中，注意的是密码的长度还是不能少于6位。（代码类似也不放出）

6) 另外，在 settingpage 中也实现了图片的选择更换：

```
private async void SelectPictureButton_Click(object sender, RoutedEventArgs e)
{
    FileOpenPicker picker = new FileOpenPicker();
    picker.ViewMode = PickerViewMode.Thumbnail;
    picker.SuggestedStartLocation = PickerLocationId.Desktop;
    picker.FileTypeFilter.Add(".jpg");
    picker.FileTypeFilter.Add(".jpeg");
    picker.FileTypeFilter.Add(".png");
    picker.FileTypeFilter.Add(".bmp");
    picker.FileTypeFilter.Add(".gif");

    StorageFile file = await picker.PickSingleFileAsync();
    if (file != null)
    {
        IRandomAccessStream stream = await file.OpenAsync(FileAccessMode.Read);
        Windows.UI.Xaml.Media.Imaging.BitmapImage bmp = new
Windows.UI.Xaml.Media.Imaging.BitmapImage();
        bmp.SetSource(stream);
        this.image.Source = bmp;
    }
}
```

7) 而当用户在进入 settingpage 的时候，我们需要将用户之前设置的个人信息还原，这时候就需要从数据库中根据用户的 username 来进行获取用户设置的信息，具体实现的代码如下：

```
using (var statement = App.conn.Prepare("SELECT nickname, email, phone, qq, wechat,
sex FROM Users WHERE username = ?"))
{
    statement.Bind(1, NewPage.user);
    while (statement.Step() != SQLiteResult.DONE)
    {
            nickname.Text = (string)statement[0];
            email.Text = (string)statement[1];
            phone.Text = (string)statement[2];
            qq.Text = (string)statement[3];
            wechat.Text = (string)statement[4];
            if (statement[5].ToString() == "male" )
                male.IsChecked = true;
            else
                female.IsChecked = true;
    }
}
```

8) 进入主界面后，首先设置 Adaptive UI。在 NewPage.xaml 里，设置以宽度为 800 划分宽屏和窄屏：

```
<VisualStateManager.VisualStateGroups>
    <VisualStateGroup>
        <VisualState x:Name="VisualStateMin0">
            <VisualState.Setters>
                <Setter Target="TheLeftGrid.(Grid.ColumnSpan)" Value="2" />
                <Setter Target="TheRightGrid.Visibility" Value="Collapsed" />
            </VisualState.Setters>
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="1" />
            </VisualState.StateTriggers>
        </VisualState>
        //……
    </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

9) 在宽屏和窄屏都能显示歌曲播放列表。歌曲播放列表通过 Models 和 ViewModels 模式来实现歌曲的储存。在 Models.SongItem 里面，设置了歌曲 id（songid）、文件名（songfilename）、歌曲名（songname）、歌手名（singer）等属性，然后在 NewPage.xaml 里两个 TextBlock 分别绑定歌曲的 songname、singer，实现数据绑定：

```
<TextBlock x:Name="songBlock" Grid.Column="0" Text="{x:Bind songname}"
    Style="{StaticResource matter_block_text}" Margin="10,0,0,0"/>
<TextBlock x:Name="singerBlock" Grid.Column="1" Text="{x:Bind singer}"
    Style="{StaticResource matter_block_text}" Margin="10,0,0,0"/>
```

10) 歌曲列表左侧，有一系列的歌曲操作按钮，分别是添加歌曲、播放、暂停、停止、上一首、下一首。分别设置点击时需要实现的功能。添加歌曲按钮需要有文件 file 方面的操作，通过 FileOpenPicker 类选取歌曲，然后设置选取类型为.mp3 格式，然后保存到 StorageFile 类：

```
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.ViewMode = PickerViewMode.Thumbnail;
openPicker.SuggestedStartLocation = PickerLocationId.PicturesLibrary;

// Users expect to have a filtered view of their folders
openPicker.FileTypeFilter.Add(".mp3");
// Open the picker for the user to pick a file
StorageFile file = await openPicker.PickSingleFileAsync();
```

11) 当歌曲文件获得后，可以通过 file.DisplayName 方法获取文件名。由于文件名一律保存为“歌曲名-歌手名”，所以通过字符串函数 Regex.Split(songfilename, "-")把文件名断开成歌曲名和歌手名。此时即可在 ViewModels 里面添加歌曲，以及把歌曲信息插入到数据库：

```
ViewModel.AddSongItem(ViewModel.AllItems.Count, songfilename, substr[0], substr[1],
    null);
var db = App.conn2;
try
{
    using (var custstmt = db.Prepare("INSERT INTO Songs (Songid, Songfilename,
        Songname, Singer, user) VALUES (?, ?, ?, ?, ?)"))
    {
        custstmt.Bind(1, ViewModel.AllItems.Count - 1);
        custstmt.Bind(2, songfilename);
        custstmt.Bind(3, substr[0]);
        custstmt.Bind(4, substr[1]);
        custstmt.Bind(5, user);
        custstmt.Step();
     }
}
catch ……
```

12) 同时，每次跳转到主页面（NewPage）时候，都会根据用户名在歌曲数据库 Songs 里查询该用户添加的歌曲，然后恢复到播放列表（以下代码在 ViewModels 里）：

```
string login_username = NewPage.user;
try
{
    using (var statement = db.Prepare("SELECT Songid, Songfilename, Songname, Singer,
        user FROM Songs"))
    {
        while (statement.Step() == SQLiteResult.ROW)
        {
            if (login_username == (string)statement[4])
            {
                _songid = (long)statement[0];
                _songfilename = (string)statement[1];
                _songname = (string)statement[2];
                _singer = (string)statement[3];
                this.allItems.Add(new Models.SongItem((int)_songid, _songfilename,
                    _songname, _singer, null));
            }
        }
    }
}
Catch……
```

13) 接下来就是歌曲的几个操作按钮的功能完善。实现思路上面已经说了，是利用 MediaElement 控件，然后设置 MediaElement.Source，然后调用 MediaElement.Play()、MediaElement.Pause()、MediaElement.Stop()等功能。首先点击每一首歌则触发 SongItem_Clicked 函数，然后把点击的歌曲的绝对路径赋予 MediaElement.Source：

```
media2.Source = new Uri("ms-appx:///Assets/" + ViewModel.SelectItems.songfilename
    + ".mp3");
```

然后再点击播放按钮，触发 broadcastButton_Click 函数，里面调用播放函数即可播放：

```
media2.Play();
```

同时点击暂停按钮或者停止按钮，则触发相应的函数，并调用一下函数，使歌曲暂停或者停止：

```
media2.Pause();
media2.Stop();
```

14) 接下来是上一首歌/下一首歌的功能。明显，当歌曲列表中播放第一首/最后一首歌时，是不能调到上一首/下一首的，此时需要根据 songid 作出判断，避免这种情况（上一首/下一首思路上一样，因此只解释上一首的代码逻辑）：

```
else if (ViewModel.SelectItems.songid == 0)
{
    var i = new MessageDialog("This is the first song now!").ShowAsync();
}
```

当正常情况时候，接下来一个最容易想到的思路是将 MediaElement.Source 改为下一首歌曲的路径，然后触发 MediaElement.Play()函数。但是事实上，这是不可行的。因为播放时候，MediaElement 的 CurrentState 为 Playing，此时 MediaElement.Play()就不会被调用，也就是说这种方法其实是起不了播放下一首的效果。

同样，另一种方法貌似也可以考虑，就是点击上一首/下一首按钮时候，先调用 MediaElement.Stop()函数，再设置 MediaElement.Source，最后触发 MediaElement.Play()函数。但是很遗憾的是，这种方法经调试也不可行。因为 MediaElement.Stop()不会改变 MediaElement 的 CurrentState，也就是后面触发 MediaElement.Play()函数时候，MediaElement 的 CurrentState 仍然为 Playing，MediaElement.Play()也不会被调用，起不了播放下一首的效果。

还有一个就是，如果歌曲在播放时候，更改该 MediaElement 的 Source 时候，歌曲播放会停止。也就是说歌曲播放的时候不能够改变 Source。

15) 为了解决上面的问题，我们采用了 3 个 MediaElement 类，设置为循环队列那样的结构，通过更改 3 者之间的关系状态，来实现每次播放时候只有一个 MediaElement 的 CurrentState 处于 Playing 状态，其他处于非 Playing 状态。同时点击上一首/下一首的时候，先通过 Stop() 函数停止已经在播放的歌曲，然后改变正在播放的 MediaElement 的前一个和后一个 MediaElement 的 Source（比如 media1 在播放，media3 即为前一首歌的储存器，media2 即为下一首歌的储存器），达到播放时候不改变播放歌曲的 MediaElement 的 Source，在播放时候不需要再设置需要播放歌曲的 Source，而是事先已经设置好的目的。

```
int playing_index = 0;
if (media1.CurrentState == MediaElementState.Playing)
{
    media1.Stop();
    playing_index = 1;
}
else if (media2.CurrentState == MediaElementState.Playing)
……

if (playing_index == 1)
{
    media3.Play();

    if (ViewModel.SelectItems.songid - 1 != 0)
    {
        media2.Source = new Uri("ms-appx:///Assets/" +
ViewModel.AllItems.ElementAt(ViewModel.SelectItems.songid - 2).songfilename +
".mp3");
    }

    media1.Source = new Uri("ms-appx:///Assets/" +
ViewModel.SelectItems.songfilename + ".mp3");

}
else if (playing_index == 2)
……
```

16) 而设置了 3 个 MediaElement 类来储存歌曲后，相应的播放和暂停、停止的函数也要做出相应的修改。要通过一个标记 lastPaused_index 记录暂停上一次暂停的是哪个 MediaElement，然后再恢复播放时候则根据那个 lastPaused_index 来播放：

```
private void pauseButton_Click(object sender, RoutedEventArgs e)
{
    //暂停选中的歌，在 SongItem_Clicked 函数已经选中某首歌曲
    if (ViewModel.SelectItems != null)
    {
```

```
        if (media1.CurrentState == MediaElementState.Playing)
        {
            media1.Pause();
            lastPaused_index = 1;
        }
        else if (media2.CurrentState == MediaElementState.Playing)
              ……
        }
           ……
    }
}
```

17) 基本上实现歌曲的相关功能后，需要通过网络访问获取歌词和专辑图片。先创建一个 HttpClient 实例对象，然后发送 GET 请求，获取返回值后进行 UTF-8 编码。将返回值转化为 json 文件，然后进行读取获得 lrc 文件。之后再发送 GET 请求获取歌词文本，最后显示出来（由于获取图片跟获取歌词类似，因此只放出获取歌词的代码）：

```
private async void GetLrc(string tel, string tel1)
{
    try
    {
        // 创建一个 HTTP client 实例对象
        HttpClient httpClient = new HttpClient();
        ……
        string getlrc = "http://geci.me/api/lyric/" + tel + "/" + tel1;
        //发送 GET 请求
        HttpResponseMessage response = await httpClient.GetAsync(getlrc);
        // 确保返回值为成功状态
        response.EnsureSuccessStatusCode();
        Byte[] getByte = await response.Content.ReadAsByteArrayAsync();
        // UTF-8 是 Unicode 的实现方式之一。这里采用 UTF-8 进行编码
        Encoding code = Encoding.GetEncoding("UTF-8");
        string result = code.GetString(getByte, 0, getByte.Length);
        JsonTextReader json = new JsonTextReader(new StringReader(result));
        string jsonVal = "", songslrc = "";

        // 获取 lrc 文件
        while (json.Read())
        {
            jsonVal += json.Value;
                if (jsonVal.Equals("lrc"))  // 读到“lrc”时，取出下一个 json token
                {
                      json.Read();
                      songslrc += json.Value;  // 该对象重载了“+=”,故可与字符串进行连接
                      break;
                }
```

```
                jsonVal = "";
            }
            //发送 GET 请求
            HttpResponseMessage response1 = await httpClient.GetAsync(songslrc);
            response1.EnsureSuccessStatusCode();
            Byte[] getByte1 = await response1.Content.ReadAsByteArrayAsync();
            Encoding code1 = Encoding.GetEncoding("UTF-8");
            string result1 = code1.GetString(getByte1, 0, getByte1.Length);
            txtb1.Text = result1;
        }
          ……
    }
}
```

18) 接下来是实现 App to app communication，主要是实现在主页面（NewPage），歌曲有 share 的菜单按钮，点击后可以分享歌曲的信息到邮件以及 OneNote，实现 APP 之间的通信功能。点击 share 菜单按钮后，注册一个 OnShareDataRequested 事件：

```
DataTransferManager.GetForCurrentView().DataRequested += OnShareDataRequested;
```

OnShareDataRequested 函数里：

```
var dp = args.Request.Data;
var deferral = args.Request.GetDeferral();

var photoFile = await StorageFile.GetFileFromApplicationUriAsync(
                                new Uri("ms-appx:///Assets/item.jpg"));
dp.Properties.Title = "Share example";
dp.Properties.Description = "A demonstration on how to share";
dp.SetText(ViewModel.SelectItems.songfilename);
dp.SetStorageItems(new List<StorageFile> { photoFile });
deferral.Complete();
```

19) 接下来再实现 Live tiles 功能。新建一个 tiles.xml 文件，设置 4 个 size 的情况。然后点击 update tiles 按钮时候，获取 xml 文件

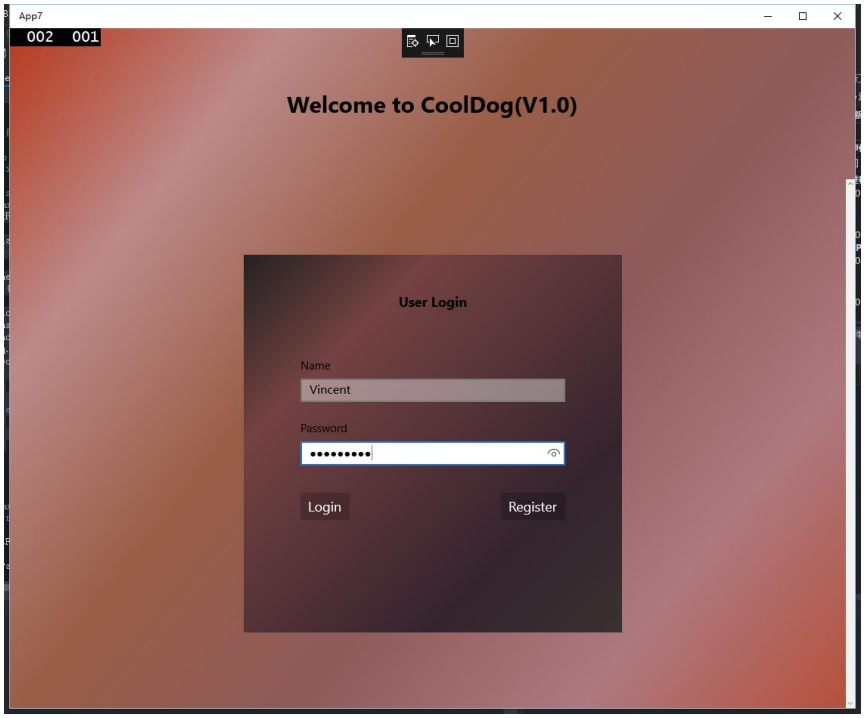
```
Windows.Data.Xml.Dom.XmlDocument tileXml = new Windows.Data.Xml.Dom.XmlDocument();
tileXml.LoadXml(File.ReadAllText("tiles.xml"));
```

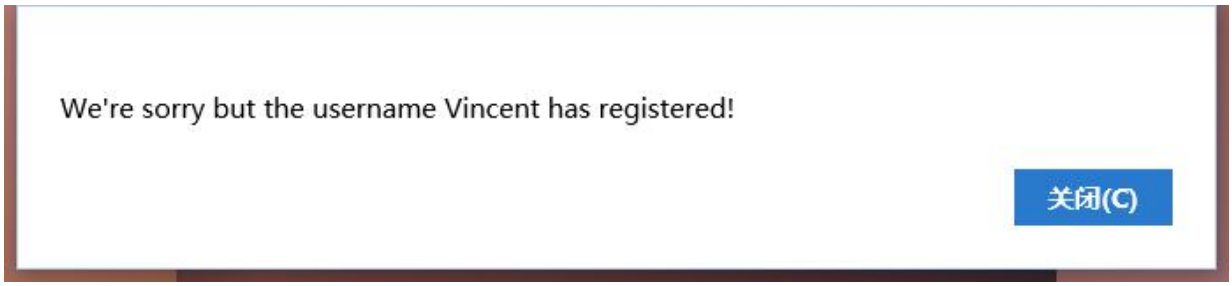
然后获取 xml 文件里面的 text 标签，再根据不同情况设置文字。最后利用 TileUpdateManager 来更新：

```
var updator = TileUpdateManager.CreateTileUpdaterForApplication();
var notification = new TileNotification(tileXml);
updator.Update(notification);
```

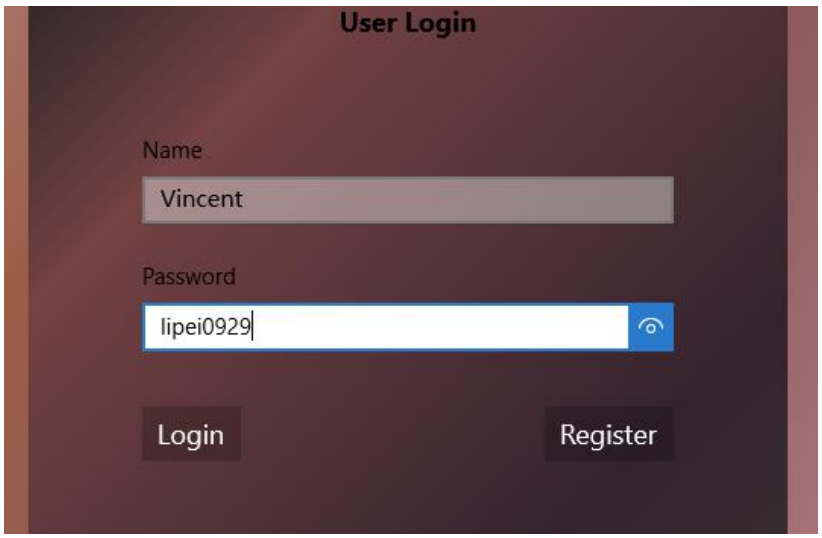
四、项目展示

1. 如图，当进入主界面的时候，如果想要注册一个数据库已经存在的用户名会弹出提示框：





2. 而当密码输入错误的时候也同样会显示登陆失败：



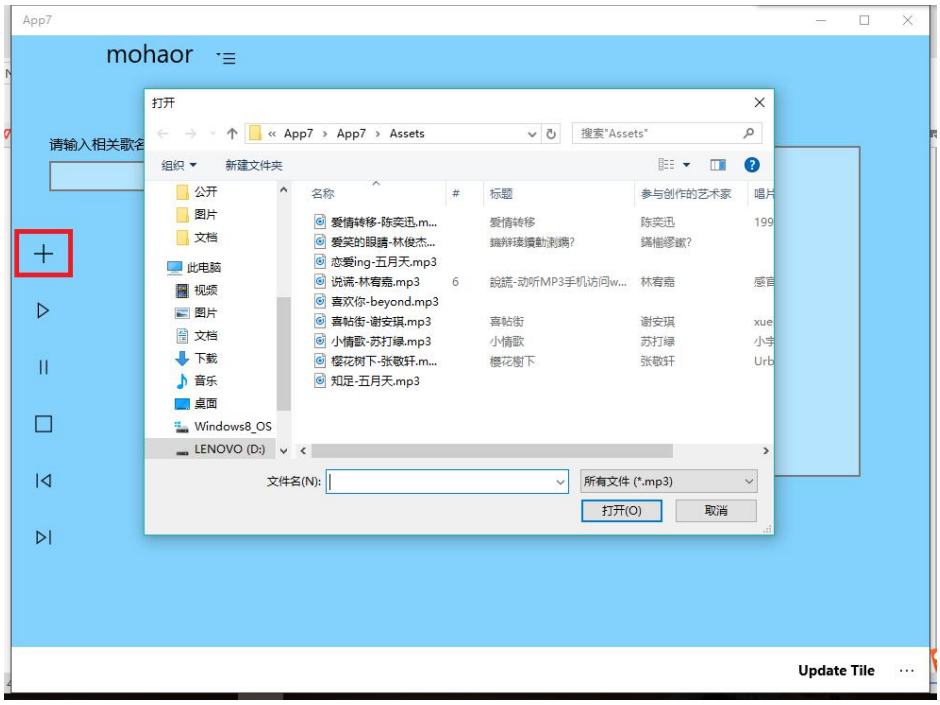
正确的密码为：

3	3	Vincent	lipei0929.	2016/05/13 11:13:23

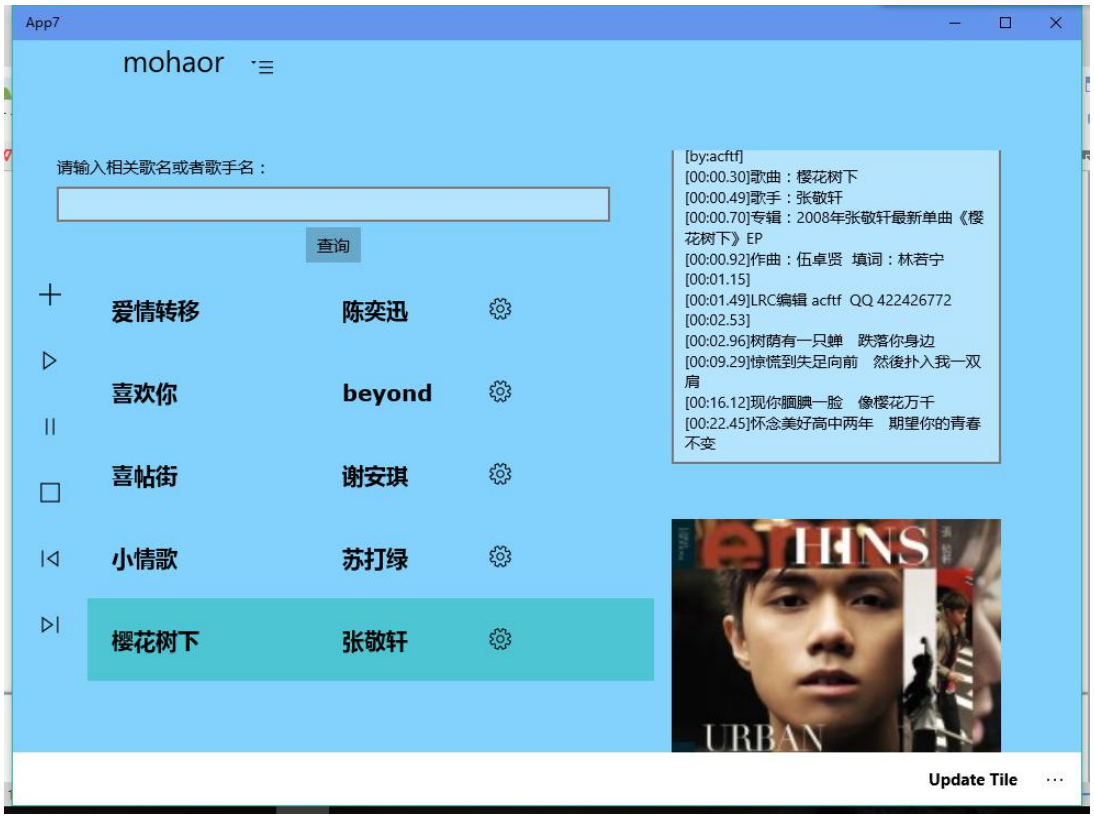
之间相差了一点= =；会弹出 login fail 的提示框：

3. 输入正确密码后，进入主页面。宽屏如下图左，窄屏如下图右：

4. 点击左侧红色框框里面的+号，能弹出一个窗口挑选歌曲：

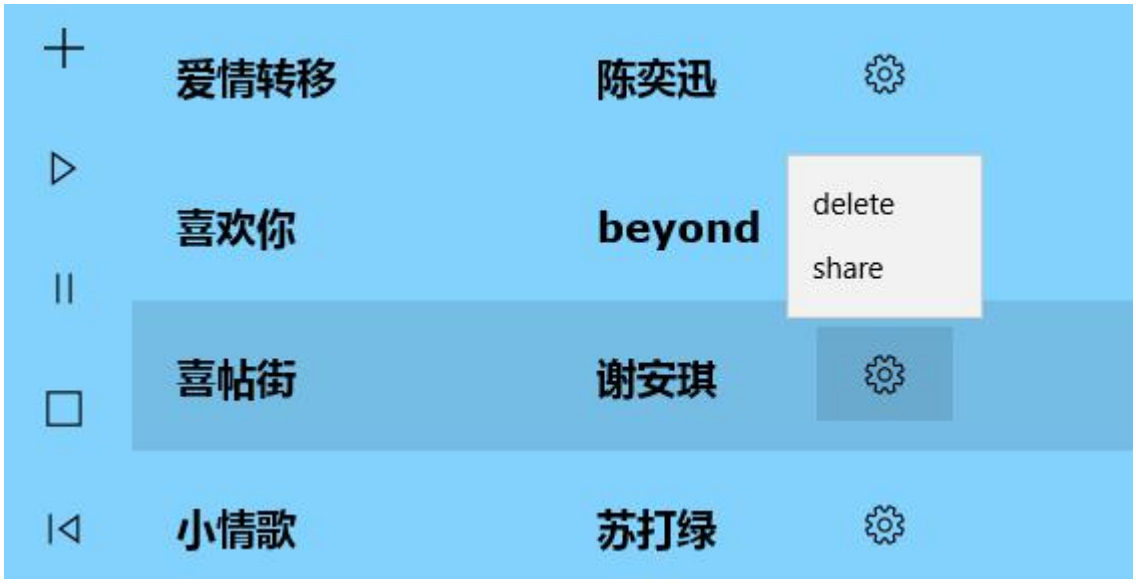


5. 把喜欢的歌曲添加进来后，能够在歌曲列表看到。点击某首歌，再点击播放按钮，就能正常播放了，右侧能显示歌词和专辑图片：

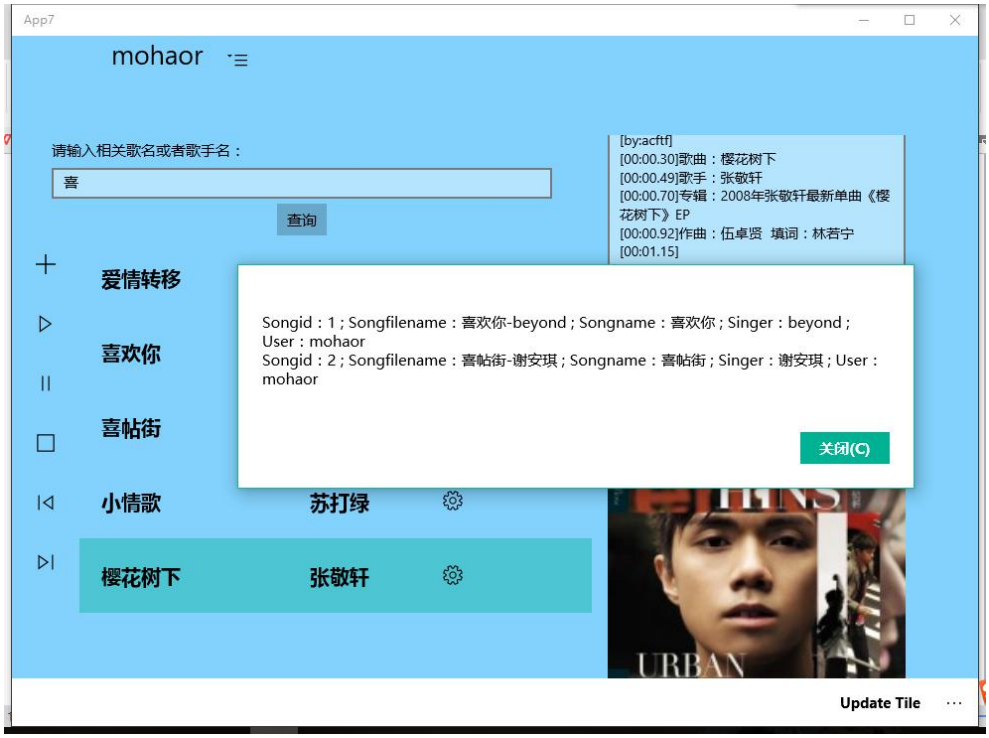


6. 左边的各个操作按钮都是能正常使用的。从上到下依次是添加歌曲、播放、暂停、停止、上一首、下一首。在切歌的时候也可以实现歌词与封面的显示（这些功能只能在视频展示）。

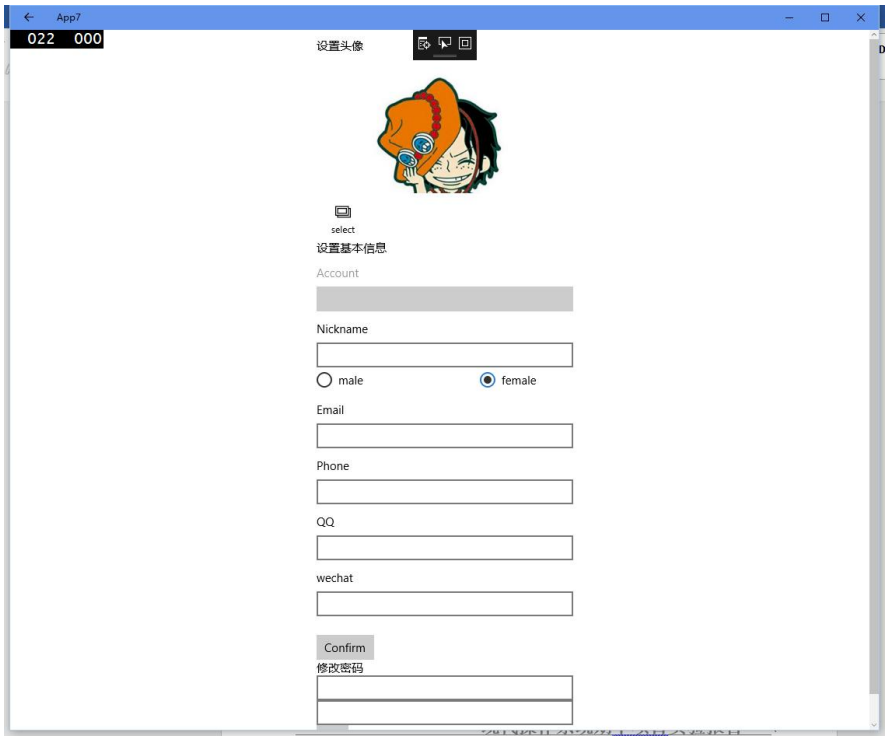
7. 可以删除歌曲：



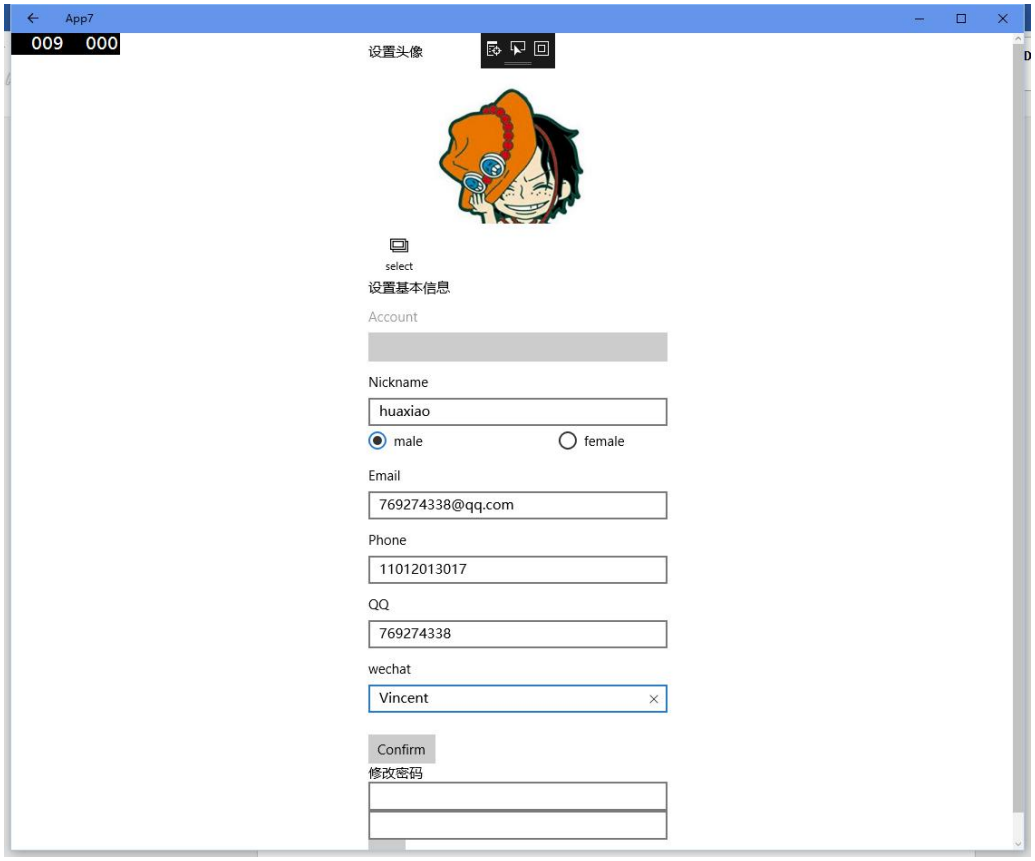
8. 在搜索那里可以输入歌手或者歌曲的相关信息，搜索歌曲列表里面的记录：



9. 通过 setting 按钮跳转到 settingpage 页面：



10. 在页面中补全个人信息：



点击 confirm 跳转回到 newpage，数据库保存用户信息：

3	3	Vincent	lipei0929.	2016/05/13 11:13:23	huaxiao	769274338@qq.com	11012013017	769274338	Vincent	male

11. 修改用户密码：

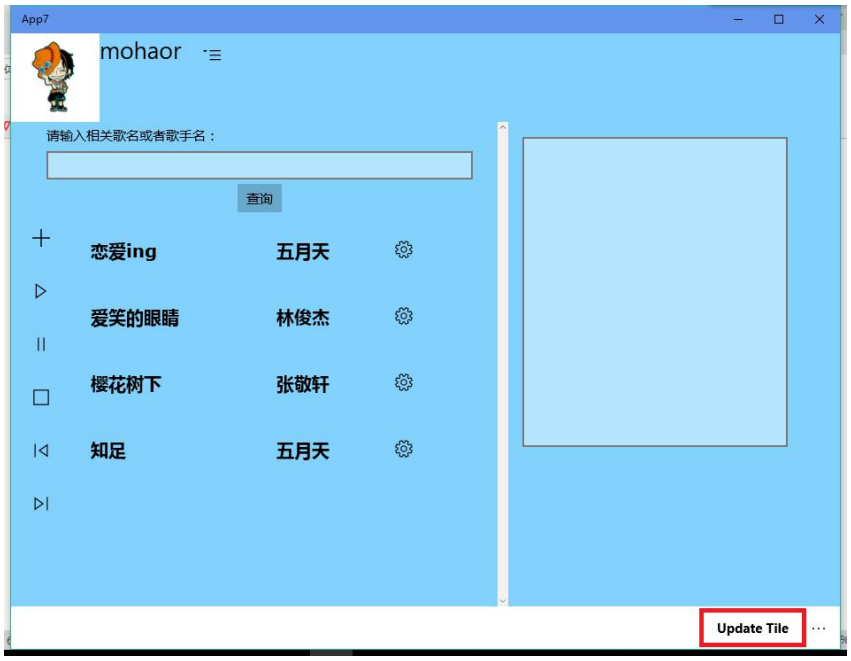
值得注意的是，如果两次输入密码不相同时，会弹出修改失败提示框，修改成功后返回到主界面，在数据库中可以看到密码更新了。

3	3	Vincent	111111	2016/05/13 11:13:23	huaxiao	769274338@qq.com	11012013017	769274338	Vincent	male

12. 点击歌曲的 share 按钮，分享到 OneNote:

然后打开 OneNote，可以发现已经收到刚分享的歌曲信息：

13. 还有最后的更新磁贴功能：点击右下方那个 update tiles 按钮后



可以看到磁贴那里显示列表中第一首歌曲的歌名。

五、项目难点及解决方案

1．具体的上面实验步骤部分也提到了，音乐播放使用的MediaElement类有很多比较隐含的问题，需要调试很久才能发现，一开始只是简单的一位换一下 MediaElement.Source，然后调用MediaElement.Play()功能就能实现播放另外的歌曲，但实际操作的时候发现不行。这方面主要是 3 大问题：Play()、Stop()函数不能改变 MediaElement 的 CurrentState 状态；CurrentState 为 Playing 时候，Play()函数不会调用；播放歌曲时候改变 Source 会使歌曲播放停止。所以使用了 3 个 MediaElement 类来通过相互之间关系来解决这个问题。

2．获取到的歌词是一个“lrc”文件，不知道怎么处理，按照 string 处理尝试的时候很巧合的显示了正确的歌词，所以得以解决；

3．在专辑封面这里想多说几句：

起初的考虑是，获取 MP3 文件自身的专辑封面，在查过很多资料后得知，需要外部引用 ID3.dll，由于引用方法复杂与原理理解不透彻导致消耗了大量时间。于是换了个思路，同样用网络访问获取“jpg”文件，当然这样做不好的地方就是多次网络访问会造成一定的延时，在获取了图片地址为 http 开头的远程图片后，用 C#图片流的方式展示图片：

以下是原始代码：

```
public partial class ShowImg : System.Web.UI.Page
{
   private string file = string.Empty;
   protected void Page_Load(object sender, EventArgs e)
    {
            //获取文件的地址参数
            file = Request.QueryString["file"].ToString();
            //以数据流的形式根据文件地址打开文件
            FileStream stream = new FileStream(file, FileMode.Open);
            //获取流的长度
            long FileSize = stream.Length;
```

```
            //定义一个二进制数组
            byte[] Buffer = new byte[(int)FileSize];
            //从流中读取字节块并将该数组写入缓冲区
            stream.Read(Buffer, 0, (int)FileSize);
            //关闭流
            stream.Close();
            //输出图片
            Response.BinaryWrite(Buffer);
            stream = null;
        }
}
```

但是该代码只能实现本地图片读取，也用过之前作业里的方法以及用 Webclient 下载图片的方法均不奏效，但是怎么也没想到，问了一些同学后，得知自带函数可能可以处理，于是最后仅用一句代码搞定所有问题：

```
image.Source = new BitmapImage(new Uri(aidimg));
```

网络访问的过程也就完成了。

六、项目总结

这个项目是从 5.1 左右开始动工的，大概是断断续续做了 2 个星期。由于一开始考虑到我们课堂上学的东西确实不是很多，能运用的知识也只是那么一点，所以只是用已经学到的知识来做一个东西的话，可能最后成品会跟我们的作业大同小异，也可能跟大多数同学做的东西差不多。所以我们小组一开始商量就打算做一些比较少同学会想到而且最后去做的东西。经过讨论，最后我们打算做一个跟音频、音乐相关的产品。这就是我们一开始的大致思路。

毕竟这个东西还是能跟以前学习过的很多东西结合起来的，比如数据库的知识，所以我们也把以前很多东西结合起来。而对于新的，与音频播放、音频操作相关的知识主要是看书，以及百度。但是对于音频相关的知识，我们在书上看的是 2010 的版本，而且百度到的东西绝大多数也是 2010 版本的。在实际操作的时候我们发现遇到了特别多问题，很多在 2010 能用的东西（比如：函数等），在 2015 版本基本用不了，或者是改动很大，完全不知道怎么使用。

举个例子吧，音频播放其实最简单的是用 MediaPlayer 类，这个 2010 版本特别容易操作，新建一个实例只需要 MediaPlayer media = new MediaPlayer()即可。但是在 2015 版本，MediaPlayer media = new MediaPlayer()，要求括号里面需要加一些参数，然后也百度不到加的参数类型是什么，弄了好久都无法解决这个最基本的问题，最后只能放弃。后来我们的用了 MediaElement 类才能新建实例。而 MediaElement 类相对于 MediaPlayer 类不是特别好用，所以要实现上下首歌的时候需要用到 3 个 MediaElement 类才能实现这样的功能。

很多 2015 版新的知识能百度到的内容真的很少。所以每次遇到问题都要一部部调试，看某个变

量的各项参数是什么。但是从这个过程中我们确实学到了很多东西，至少是在一种完全找不到现成的解决方案的时候，通过我们自己的努力去把所有遇到的问题解决。所以这个项目也让我们感到很满意。

现在这个 APP 应该是没有什么 bug 的了（至少我们各种测试也还没发现什么问题）。不过觉得还有更多的与音频相关的功能还可以实现，比如设置音量、设置歌曲播放进度等。这些在书上貌似也可以解决，不过因为时间问题，我们没有实现那么多的功能。