

MUSIC APPLICATION

* **Faculty: Ho Hoan Kiem**
* **Batch:**
* **Team Member:**
  + - * **Truong Le Gia Luat**
      * **Vo Thanh Tung**

**Ho Chi Minh City, Sep 10 2012**

REVIEW 1

Problem Definition

1. **The Object Of The Project**

This project will provide a music application. This application allow everyone to listen to music. For user register an account on music application, he can manage many playlists where contain many favorite songs.

Customer Request   
 Specification

1. **Functional Requirements**

Following is a list of functionalities of the system. More functionalities that you find appropriate can be added to this list.

1. User can listen to the songs on the application without needing to register account
2. User can register the account on the music application.
3. User must input username and password to login on the music application.
4. User can download the song from the music application.
5. Each account have many playlists to save the favorite songs in many genres.
6. Each account can upload many songs to the music application.
7. User can search song by inputing title of song.
8. User can see the artist and get list of songs of that artist.

# **Hardware/ Software Requirements**

By adding new technologies such as WPF, WCF and 3-layer architecture, the WPF form becomes more interactive with users, security and easily updated.

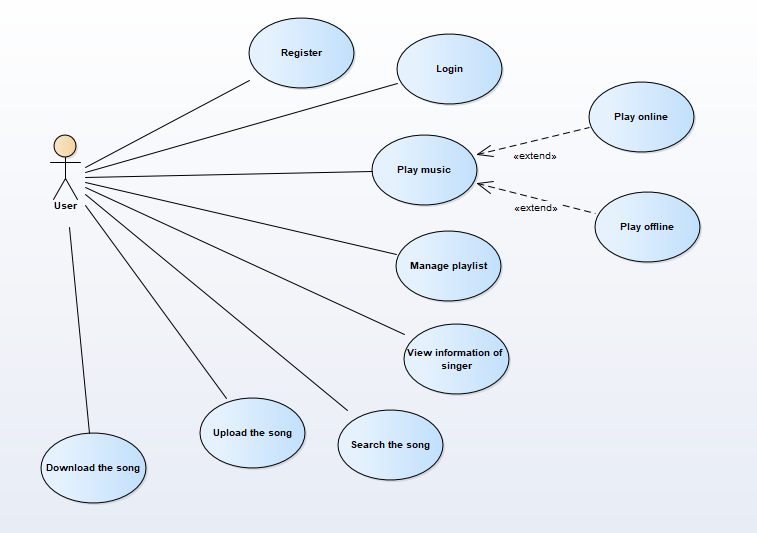
1. **System Requirement**

* **For Users**
  + **Hardware**
    - CPU: Pentium IV 2.0 GHz or higher
    - RAM: 512 MB or higher
  + **Software**
    - OS: Windows 7/XP Professional/Server 2003…
    - Browser: Internet Explorer 8.0 or higher
* **For Software Developers**
  + **Hardware**
    - CPU: Pentium IV 2.0 GHz or higher
    - RAM: 512 MB or higher
  + **Software**
    - OS: Windows XP Professional/Server 2003…
    - Environment: Microsoft .NET Framework 3.5
    - Database Management System: Microsoft SQL Server 2017
    - IDE: Microsoft Visual Studio 2017

REVIEW 2

**USE CASE DIAGRAM**

USER



**USE CASE DESCRIPTION**

1. **Actor:**

* **Users:** They are guests and members. ***Guest*** is the person that uses this application but doesn’t register. ***Member*** is the person that uses this application and register to become a member of us.
* **Administrators :** The person(s) that manage and maintain this application.

1. **Use case:**

|  |  |  |
| --- | --- | --- |
| **Usecase** | Register | |
| **Short Description** | Register account to become a member of the music application | |
| **Pre-Conditions** |  | |
| **Post-Conditions** |  | |
| **Main flow** | User actions:  1. Guest must provide mandatory information :   * Username. * Password. * Confirm Password. * Email. * Birthday.   2. Guest provides supplementary information: FullName, PhoneNumber, Address (can no input)  3. Guest clicks on Register button.  [Alternative 1] | System responses:  4. System checks the information which guest provided  [Exception 1]  5. System creates new account for guest that registers to this application and notifies message pop-up: “You register the account successfully!” |
| **Alternative flow(s)** | User actions:  - User click “Close” button on “register” pop-up | System responses:  - System turn off “Register” pop-up |
| **Exception(s)** | User actions:  [Exception 1]  - If the mandatory information that guest  provided, is missed or not exactly, system notifys the error. All cases are :  + User lefts “UserID” text box blank.  + UserID has already existed.  + Length of userID is too long compared to condition  + Actor lefts “Password” text box blank.  + Length of Password is too long compared to condition  + Actor lefts “Confirm Password” text box blank.  + Password and confirm password don’t match  + User lefts “Birthday” textbox blank.  + Birthday doesn’t match with pattern (dd/mm/yyyy).  + User lefts ‘Email’ text field blank  + Email doesn’t match with pattern.  - If the suplementary information that guest provided, is invalid form, system nofitfys the error. All cases are :  + Input “Phone number” os not in numeric or length of phone number less than 10 and more than 11. | System responses:  System displays message on “Register” pop-up:  - UserID can’t be blank!.  - UserID has existed!.  - Length of userID is out of range!.  - Password must be more than 8 character!.  - Length of userID is out of range!.  Length of userID is less than 30  - Confirm Password can’t be blank!.  - Confirm Password and Password are not the same!.  - Birthday cannot be blank!.  - Birthday is invalid!.  - Email cannot be blank!.  - Email is invalid!.  - The phone number should be numeric only and must be from 10 to 11 digits. |

|  |  |  |
| --- | --- | --- |
| **Use case** | Login | |
| **Short Description** | Login | |
| **Pre-Conditions** |  | |
| **Post-Conditions** |  | |
| **Main flow** | Actor Actions:  1. Actor clicks ‘Login’ link in Home page right top corner of the application.              3. Actor inputs login’s information and clicks ‘Login’ button on popup. [Alternatiove 1] | System Responses:  2. System redirect to login pop-up and displays Login form with the following control:  - ‘Username’ text field  - ‘Password’ text field  - ‘Login’ button  4. System checks the formal validate  [Exception 1]  5. System send the information to the server and get the result of Login action.  5. System check the result and turn popup off. Replace ‘Login’ link with username and add ‘Logout’ button next to it.  [Exception 2] |
| **Alternative flow(s)** | Actor Actions:  [Alternative 1]  1. Actor click "Close" button. | System Responses:  - System turn off “Login” pop-up |
| **Exception(s)** | Actor Actions:  [Exception 1]  1. Actor lefts ‘Username’ text box blank.  2. Actor lefts ‘Password’ text box blank.  3. Input Username length not in range 8 – 30 characters.  4. Input Password length not in range 8 – 30 characters.  [Exception 2]  1. Input Username or Password wrong | System Responses:  System displays message to Actor:  - ‘Username cannot be blank’.  - ‘Password cannot be blank’’.  - ‘The Username length must in rang 8 – 30 characters’.  - ‘The Password length must in rang 8 – 30 characters’.  - ‘The username or password is wrong’ |

|  |  |  |
| --- | --- | --- |
| **Use case** | Manage playlist | |
| **Short Description** | Manage playlist | |
| **Pre-Conditions** | Guest login as user | |
| **Post-Conditions** |  | |
| **Main flow** | User actions:  1. User turn on application  3. User click “My music” button  5. User click “Playlist” tag  7. User click “Add playlist” button  9. User input Name of playlist and click “Create new”  [Alternative 1] | System Response:  2. System show the music application form  4. The application will become Playlist window include:  - “Song” tag  - “Album” tag  - “Artist” tag  - “Playlist” tag  6. System show “Playlist” form include:  - “Add playlist” button  8. System show form include:  - “Name of playlist” text box  - “Cancel” button  - “Create new” button  10. System checks the information which user provided  [Exception 1]  11. System send data to server  12. Server insert into database and create a playlist then return the result to the system.  13. System update the playlist form again. |
| **Alternative flow(s)** | User actions:  - User click “Cancel” button on | System Response:  - System turn off form |
| **Exception(s)** | User actions:  - User left “Name of playlist” textbox blank | System Response:  System displays message to Actor:  - Name of playlist cannot blank! |

|  |  |  |
| --- | --- | --- |
| **Use case** | Play music | |
| **Short Description** | Play music | |
| **Pre-Conditions** |  | |
| **Post-Conditions** |  | |
| **Main flow** | Actor actions: 1. Actor click “Song” button is on left side of the application  3. Actor click “Play” button | System responses:  2. System show a songs list, each song include:  + “Song” image  + “Name of Song” label  + “Name of artist” label  + “Play” button  + ”Add playlist” button  + “Download” button  4. System redirect to playing song form  5. System send information of the song to the server.  6. Server get the []byte of song and send back to the system.  7. System play the song. |
| **Alternative flow(s)** |  |  |
| **Exception(s)** |  |  |

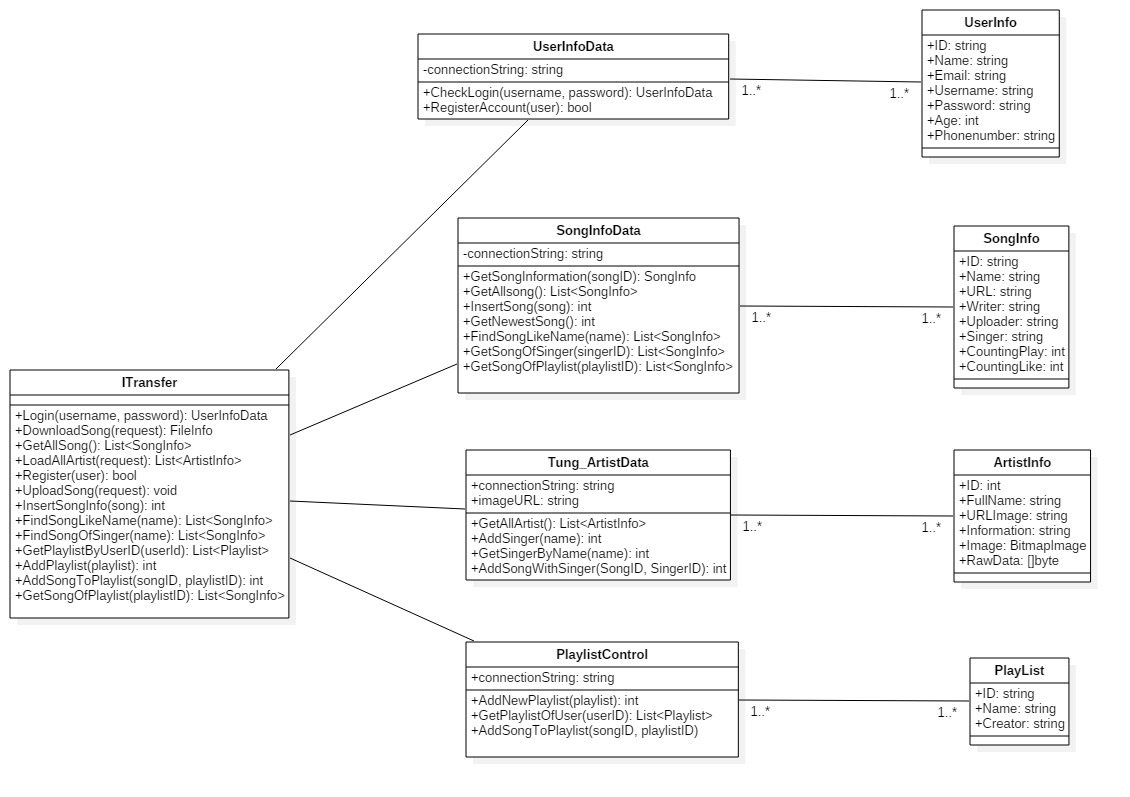
|  |  |  |
| --- | --- | --- |
| **Use case** | View information of singer | |
| **Short Description** | View information of singer | |
| **Pre-Conditions** |  | |
| **Post-Conditions** |  | |
| **Main flow** | Actor actions: 1. Guest click on singer name or singer profile picture.  3. Actor click “Get song” button | System responses:  2. Main windows become the windows of singer’s information include:  + Singer’s image  + “Singer’s name” label  + “Get song” button  4. System redirect to song control with the list of song of that singer. |
| **Alternative flow(s)** |  |  |
| **Exception(s)** |  |  |

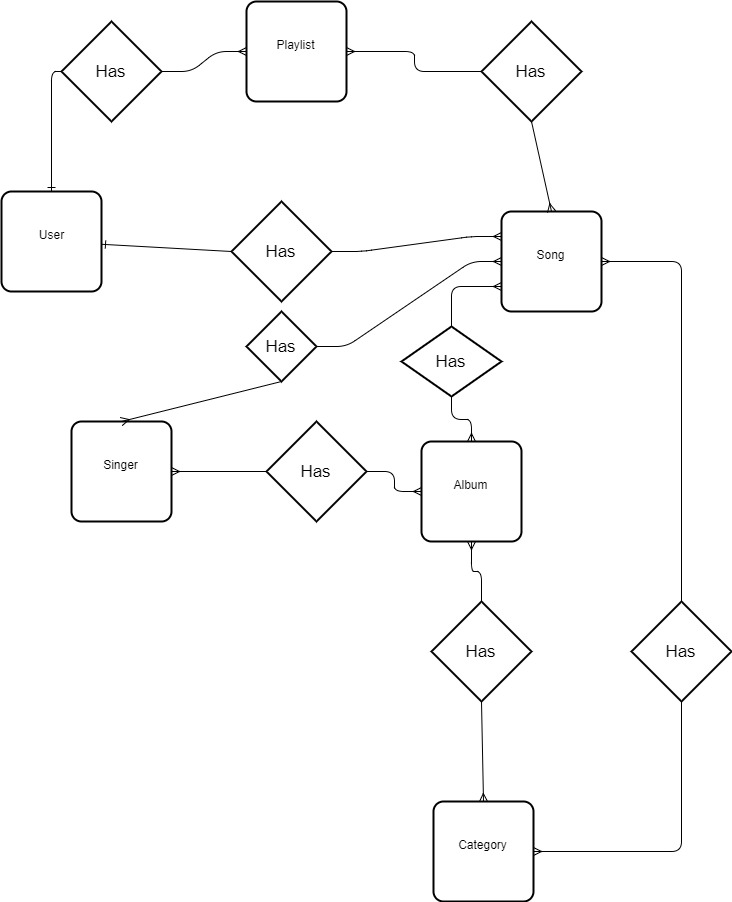
|  |  |  |
| --- | --- | --- |
| **Use case** | Search the song | |
| **Short Description** | Search the song singer | |
| **Pre-Conditions** |  | |
| **Post-Conditions** |  | |
| **Main flow** | Actor actions: 1. User input a part of song name into search text box then press “Enter”. | System responses:  2. System get the information and send to the server.  [Exception 1]  3. Server search the list of song like the name then return the result to the system.  4. System show the song control form of that result. |
| **Alternative flow(s)** |  |  |
| **Exception(s)** | [Exception 1]  1. Search text box blank | Show the message box “Search value cannot be blank”. |

|  |  |  |
| --- | --- | --- |
| **Use case** | Upload the song | |
| **Short Description** | Upload the song | |
| **Pre-Conditions** | Guest login as the Member | |
| **Post-Conditions** |  | |
| **Main flow** | Actor actions: 1. User must provide the following information:  + Song name  + Singer  + Lyric  + The file of song ( .mp3 type)  - User press upload button then a message box show that “Upload song”. | System responses:  2. System get the information from the form and the user id and send to the server.  [Exception 1]  3. Server insert song information and return new song id to the system.  4. System send the file with the song id to server.  5. System show message box show “Upload successfully.”. |
| **Alternative flow(s)** |  |  |
| **Exception(s)** | [Exception 1]  1. Song name blank  2. Singer blank  3. The file of song blank  4. File of song is not .mp3 type | 1. Show the message box “Song name value cannot be blank”.  2. Show the message box “Singer cannot be blank”.  3. Show message box “Please choose the file”.  4. Show message box “Please select a .mp3 type file” |

|  |  |  |
| --- | --- | --- |
| **Use case** | Download the song | |
| **Short Description** | Upload the song | |
| **Pre-Conditions** | Guest login as the Member | |
| **Post-Conditions** |  | |
| **Main flow** | Actor actions: 1. User click download button in song window  3. User choose the URL to save file then click “Ok”. | System responses:  2. A file picker pop-up for user to choose where to save file  4. System get the information of the song and send to the server.  [Alternative 1]  5. Server send the file of that song id to the system.  6. System save the received file to the URL. |
| **Alternative flow(s)** | [Alternative 1]  1. User not choose the the URL | 1. System show the song list again. |
| **Exception(s)** |  |  |

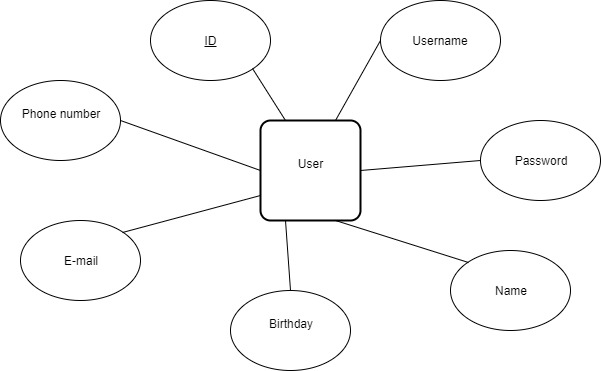
**CLASS DIAGRAM**



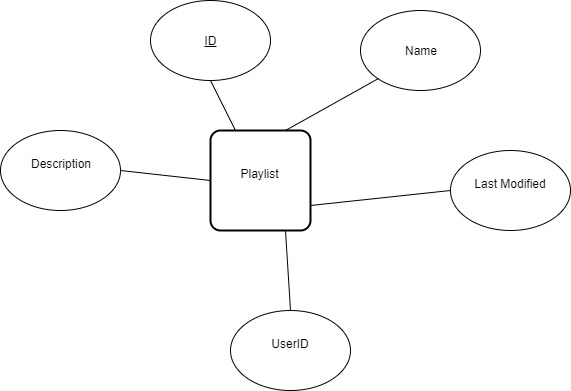
**ENTITY RELATIONSHIP DIAGRAM (ERD)**

**ENTITY DESCRIPTION**

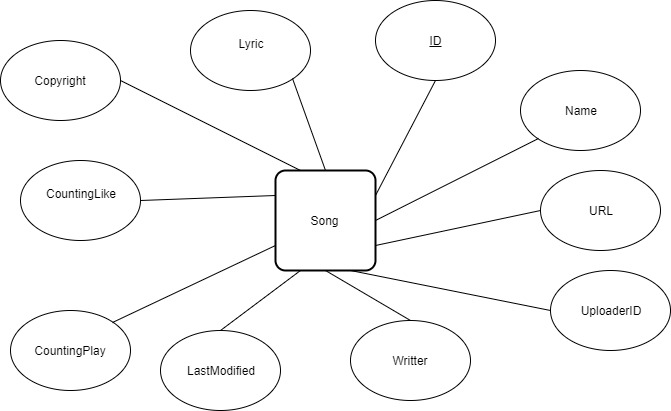
**1/ User**

****

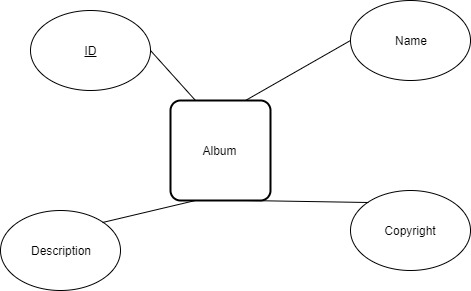
**2/ Playlist**

****

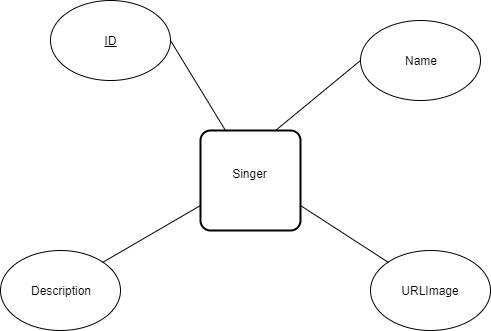
**3/ Song**

****

**4/ Album**

****

**5/ Singer**

****

**6/ Category**

****

**TABLE DESIGN**

**1/ User**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Column Name** | **Data Type** | **Size** | **Key** | **Check** | **Description** |
| 1 | ID | int |  | Identity,  Primary  key | Not Null |  |
| 2 | Username | Nvarchar | 20 | Unique  key | Not Null |  |
| 3 | Password | Nvarchar | 20 |  | Not Null |  |
| 4 | Name | Nvarchar | 50 |  | Not Null |  |
| 5 | Birthday | Date |  |  |  |  |
| 6 | E-mail | Nvarchar | 50 |  |  |  |
| 7 | Phone number | Nvarchar | 11 |  |  |  |

**2/ Song**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Column Name** | **Data Type** | **Size** | **Key** | **Check** | **Description** |
| 1 | ID | int |  | Identity,  Primary  key | Not Null |  |
| 2 | Name | Nvarchar | 20 |  | Not Null |  |
| 3 | URL | Nvarchar | 50 |  |  |  |
| 4 | UploaderID | int |  | Foreign  Key  User.ID | Not Null |  |
| 5 | Writter | Nvarchar | 50 |  | Not Null |  |
| 6 | LasModified | Datetime |  |  | Not Null |  |
| 7 | CountingPlay | int |  |  |  |  |
| 8 | CountingLike | int |  |  |  |  |
| 9 | Copyright | Nvarchar | 50 |  |  |  |
| 10 | Lyric | Nvarchar | 50 |  |  |  |

**3/ Singer**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Column Name** | **Data Type** | **Size** | **Key** | **Check** | **Description** |
| 1 | ID | int |  | Identity,  Primary  key | Not Null |  |
| 2 | Name | Nvarchar | 50 |  | Not Null |  |
| 3 | URLImage | Nvarchar | 50 |  | Not Null |  |
| 4 | Description | Nvarchar | 50 |  |  |  |

**4/ Playlist**

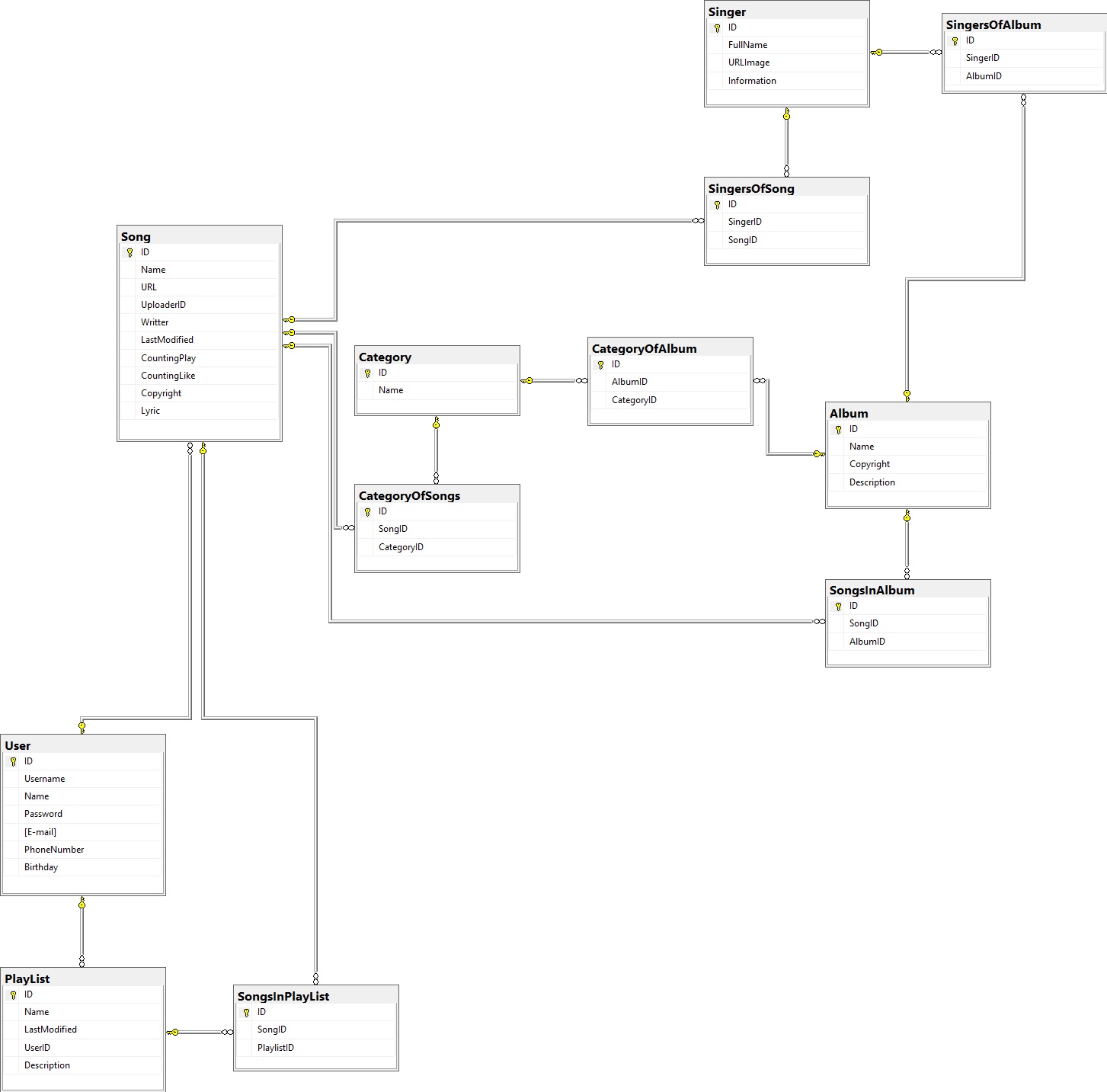
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Column Name** | **Data Type** | **Size** | **Key** | **Check** | **Description** |
| 1 | ID | int |  | Identity,  Primary  key | Not Null |  |
| 2 | Name | Nvarchar | 20 |  | Not null |  |
| 3 | LastModified | Datetime |  |  |  |  |
| 4 | UserID | int |  | Foreign  key  User.ID | Not null | Creator’s UserID |
| 5 | Description | Nvarchar | 50 |  |  |  |

**5/ Category**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Column Name** | **Data Type** | **Size** | **Key** | **Check** | **Description** |
| 1 | ID | int |  | Identity,  Primary  key | Not Null |  |
| 2 | Name | Nvarchar | 50 |  | Not null |  |

**6/ Album**

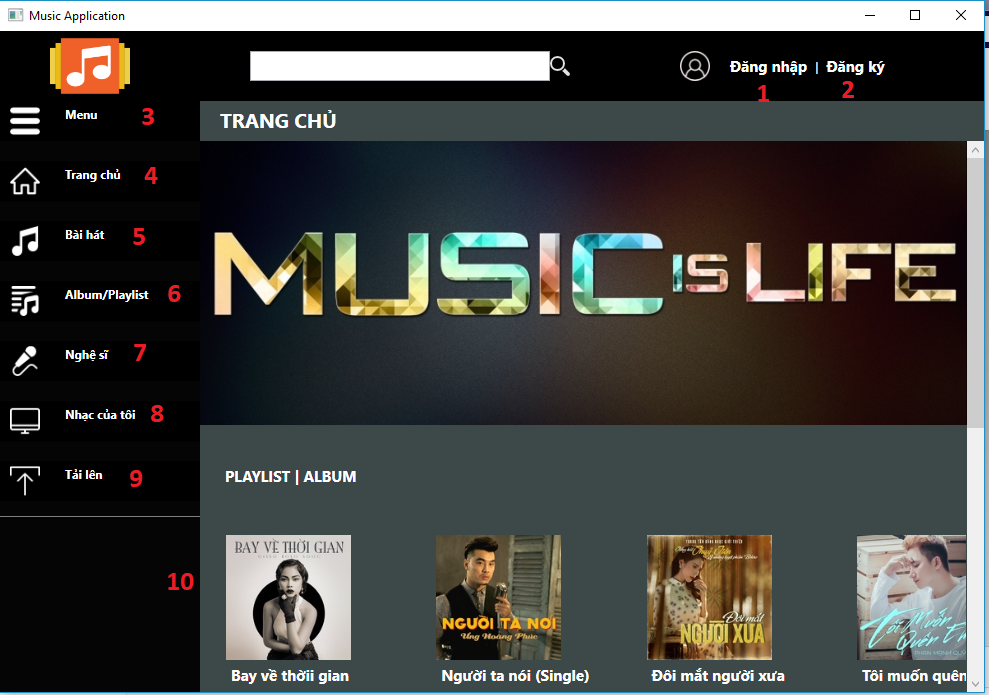
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Column Name** | **Data Type** | **Size** | **Key** | **Check** | **Description** |
| 1 | ID | int |  | Identity,  Primary  key | Not Null |  |
| 2 | Name | Nvarchar | 50 |  | Not Null |  |
| 3 | Copyright | Nvarchar | 50 |  |  |  |
| 4 | Description | Nvarchar | 50 |  |  |  |

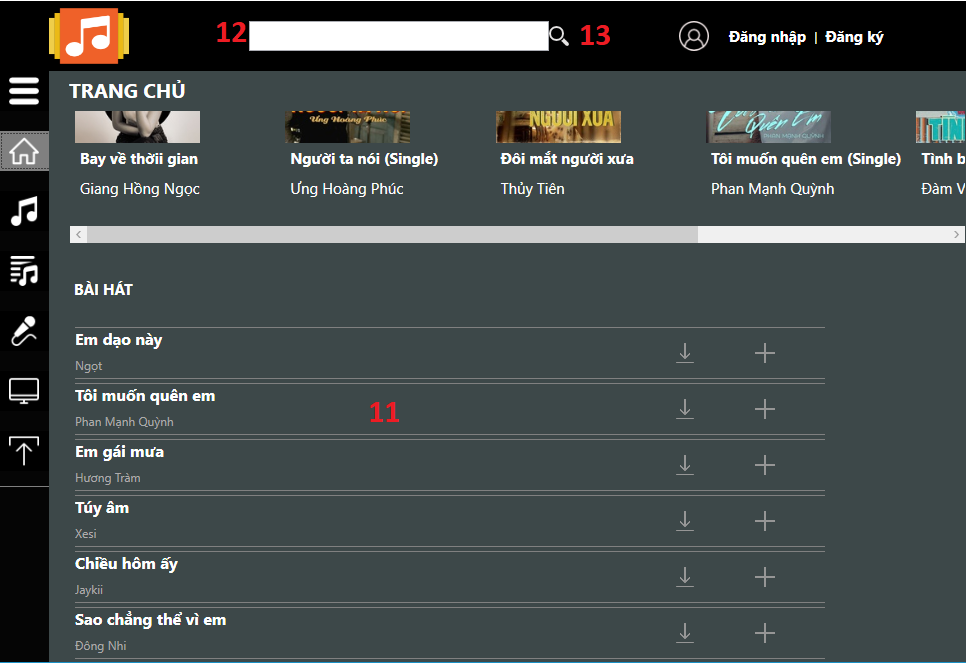
**Database Diagram**

REVIEW 3

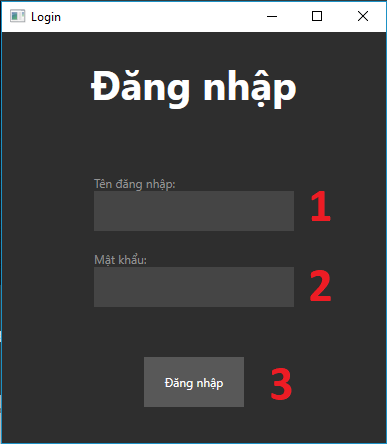
**INTERFACE DESIGN**

**1. Home page**

****

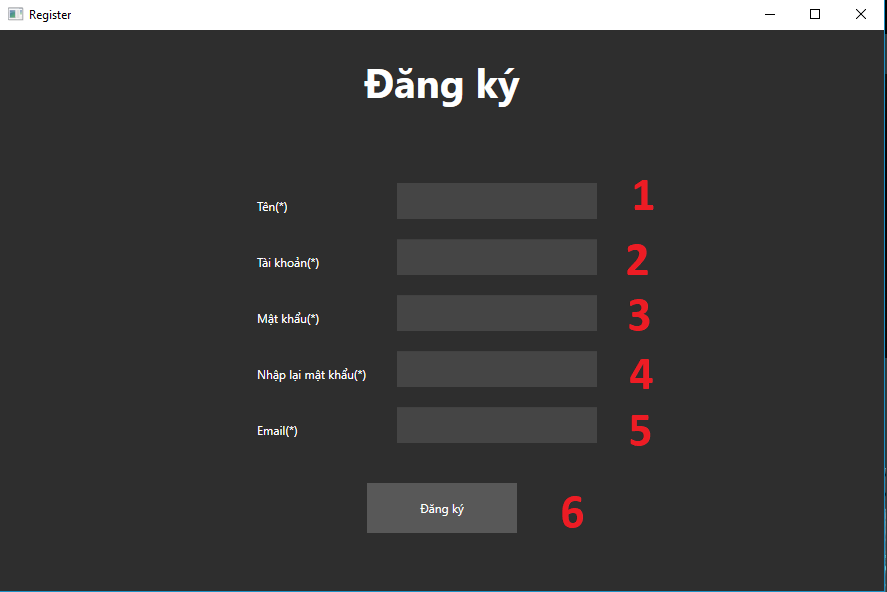
****

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Name** | **Type** | **Event** | **Validation** | **Description** | **Status** |
| 1 | tbLogin | TextBlock | MouseLeftButtonDown |  | Click to show login form |  |
| 2 | tbRegister | TextBlock | MouseLeftButtonDown |  | Click to show register form |  |
| 3 | btnMenu | Button | Click |  | Click to show name of button on menu |  |
| 4 | btnHome | Button | Click |  | Click to show Home page |  |
| 5 | btnSong | Button | Click |  | Click to show Song page |  |
| 6 | btnAlbum | Button | Click |  | Click to show Album page |  |
| 7 | btnArtist | Button | Click |  | Click to show Artist page |  |
| 8 | btnMyPlaylist | Button | Click |  | Click to show MyPlaylist page |  |
| 9 | btnUpload | Button | Click |  | Click to show Upload page |  |
| 10 | lvAbum | ListView |  |  | List view of Ablum |  |
| 11 | lvSong | ListView |  |  | Listview of Song |  |
| 12 | tBoxSearch | TextBox |  | Not empty when click Search button |  |  |
| 13 | btnSearch | Button | Click |  | Click to find name of song |  |

**2. Login form**

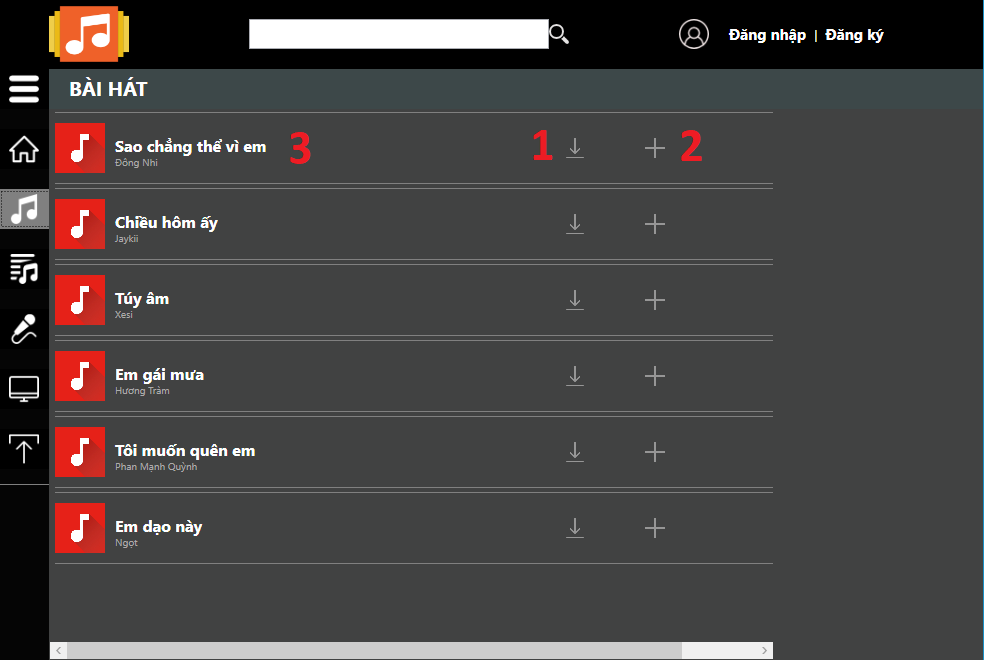
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Name** | **Type** | **Event** | **Validation** | **Description** | **Status** |
| 1 | tBoxUsername | TextBox |  | Length >= 6 | Username of user |  |
| 2 | tBoxPassword | TextBox |  | Length >= 8 | Password of user |  |
| 3 | btnLogin | Button | Click |  | Click to login on the application |  |

**3. Register form**

****

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Name** | **Type** | **Event** | **Validation** | **Description** | **Status** |
| 1 | tBoxName | TextBox |  | Length > 0 | Name of user |  |
| 2 | tboxUsername | TextBox |  | Length >= 6 | Username of user |  |
| 3 | tboxPassword | TextBox | Click | Length >= 8 | Password of user |  |
| 4 | tboxConfirmPassword | TextBox |  | ConfirmPassword == Password | Confirm Password |  |
| 5 | tboxEmail | TextBox |  | Match Pattern:  \w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\* | Email of user |  |
| 6 | btnRegister | Button |  |  | Click to register to the application |  |

**4. Song page**

****

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Name** | **Type** | **Event** | **Validation** | **Description** | **Status** |
| 1 | tBoxName | TextBox |  | Length > 0 | Name of user |  |
| 2 | tboxUsername | TextBox |  | Length >= 6 | Username of user |  |
| 3 | tboxPassword | TextBox | Click | Length >= 8 | Password of user |  |
| 4 | tboxConfirmPassword | TextBox |  | ConfirmPassword == Password | Confirm Password |  |
| 5 | tboxEmail | TextBox |  | Match Pattern:  \w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\* | Email of user |  |
| 6 | btnRegister | Button |  |  | Click to register to the application |  |

**5. Album page**

**6. Artist page**