Department of Computer Science

Software Project Proposal

|  |  |  |
| --- | --- | --- |
| **Course** OBJECT ORIENTED PROGRAMMING 2 | | **Group No.** |
| **Student ID** | **Name** | **Section** |

22-46280-1 Maeed Ahammed N

21-44853-2 AMIT HASAN RUHIN N

22-49544-3 MD RIZVI KARIM OHY N

21-45224-2 MD. HIZBUL BASHAR SHOURAV N

**Project Title:**

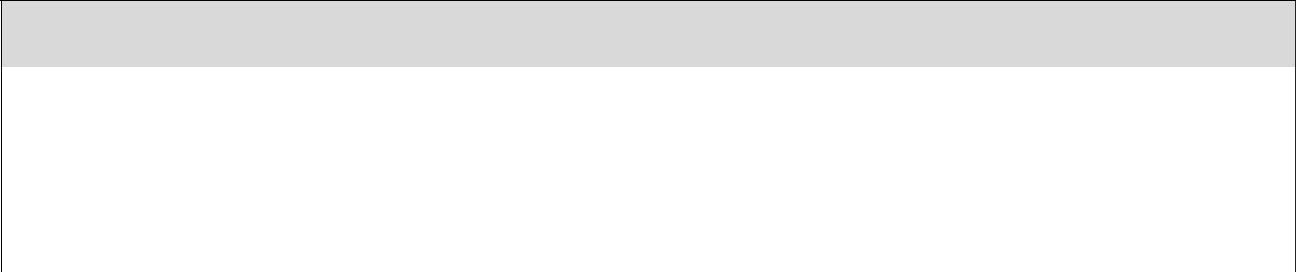
|  |
| --- |
| Music Player (HARMONY) |

**Project Description:**

A central focus for your listening experience is provided by an easy-to-use music player system with a Windows Forms user interface, designed with C# and OOP concepts. Using the built-in search and discovery tools, you may import and arrange your music collection or find new tunes. Make your own playlists to suit any mood, then take charge with simple playback controls like volume, pause, and play. Use an equalization to adjust the sound and customize the UI to your taste. With the help of this music player, you can organize and control your whole audio library in one location.

|  |  |  |  |
| --- | --- | --- | --- |
| • | Entity Classes | • | n-Tier Architecture (optional) |
| * At least 2 types of Users | | • | Separate Login Table |
| • | Database CRUD operations | • | Implement OOP Principles |
| * All the Forms MUST be connected. | | * Application Layer (Form Design) | |
| • | Database Connection Class | • | One Complete Repository. |

* Mention all the **Entities** and **attributes for each Entity** clearly in your Project proposal.



**Project Requirements:**

* Specify how many roles there are in your system and what are their individual functionalities.
* You must provide your Project Proposal following the format given below.
* Use only the first page of this document as a cover page for your Project proposal.
* **Entity Classes:** The system will define classes representing real-world entities like User, Playlist, Song, Artist, Album etc. These classes will encapsulate data and provide methods for manipulation.

**REQUIREMENTS:**

* **User Types:** The system will support at least two types of users: Artist, Listener.
* **Separate Login Table:** A dedicated table will store user credentials (username, password) for secure login functionality.
* **Database CRUD Operations:** The application will implement functionalities to Create, Read, Update, and Delete data within the connected database.
* **OOP Principles:** The development will adhere to OOP principles like Encapsulation, Inheritance, Polymorphism, and Abstraction.
* **Form Connectivity:** All forms within the application will be seamlessly connected to allow users to navigate and interact with different functionalities.
* **Application Layer (Form Design):** Visually appealing and user-friendly Windows Forms will be designed to provide a smooth user experience.
* **Database Connection Class:** A dedicated class will handle database connection and interaction, simplifying data access.
* **One Complete Repository:** The system will implement a repository pattern to manage data access logic for one specific entity demonstrating this design pattern.

# REQUIREMENT SPECIFICATION:

* User Stories

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Actor** | **User Story** |
| Admin | Admin | As an Admin, I want to manage users in the full application, so that I can ensure the security and proper functioning of the system. |
| Register User | Artist, Listener, Visitor | The registered user wants to create a new playlist to organize their music collection. |
| Premium User | Artist, Listener | The premium user wants to access high-quality audio streaming within the music player system. |
| Artist | Artist | The artist wants to upload their music content to the music player system for distribution and streaming. |
| Search For Music | Admin, Register User, Premium User | The registered user wants to find specific songs, artists, or albums within the music player system's library. |
| Download Songs | Admin, Register User, Premium User, Artists | The user wants to download songs for offline playback within the music player system |
| Create Playlist | Register User, Premium User, Artists | The registered user wants to create a new playlist to organize their music collection. |
| Play Offline | Admin, Register User, Premium User, Artists | The user wants to listen to their music library even without an internet connection |

|  |  |  |
| --- | --- | --- |
| Login and Logout from the system | Admin, Register User, Premium User, Artists | The user wants to securely access their account within the music player system and log out when finished. |
| Update profile | Admin, Register User, Premium User, Artists | The registered user wants to modify their account information within the music player system. |
| Manage User | Admin | The administrator wants to manage user accounts within the music player system, ensuring a healthy user base and system functionality. |
| Manage Subscription | Admin, Register User, Premium User, Artists | The registered user wants to manage their subscription plan within the music player system. This use case can be adapted for Admins managing system-wide plans as well. |
| Manage Full Application | Admin | I want to manage users in the full application, so that I can ensure the security and proper functioning of the system. |
| Alter Account Details | Admin, Register User, Premium User, Artists | The registered user wants to modify their account information within the music player |
| Upload Music | Admin, Artists | The artist wants to upload their music content to the music player system for distribution and streaming. |
| Manage Music | Admin, Register User, Premium User, Artists | * Manages their uploaded music. Manages their music library within the music player system |

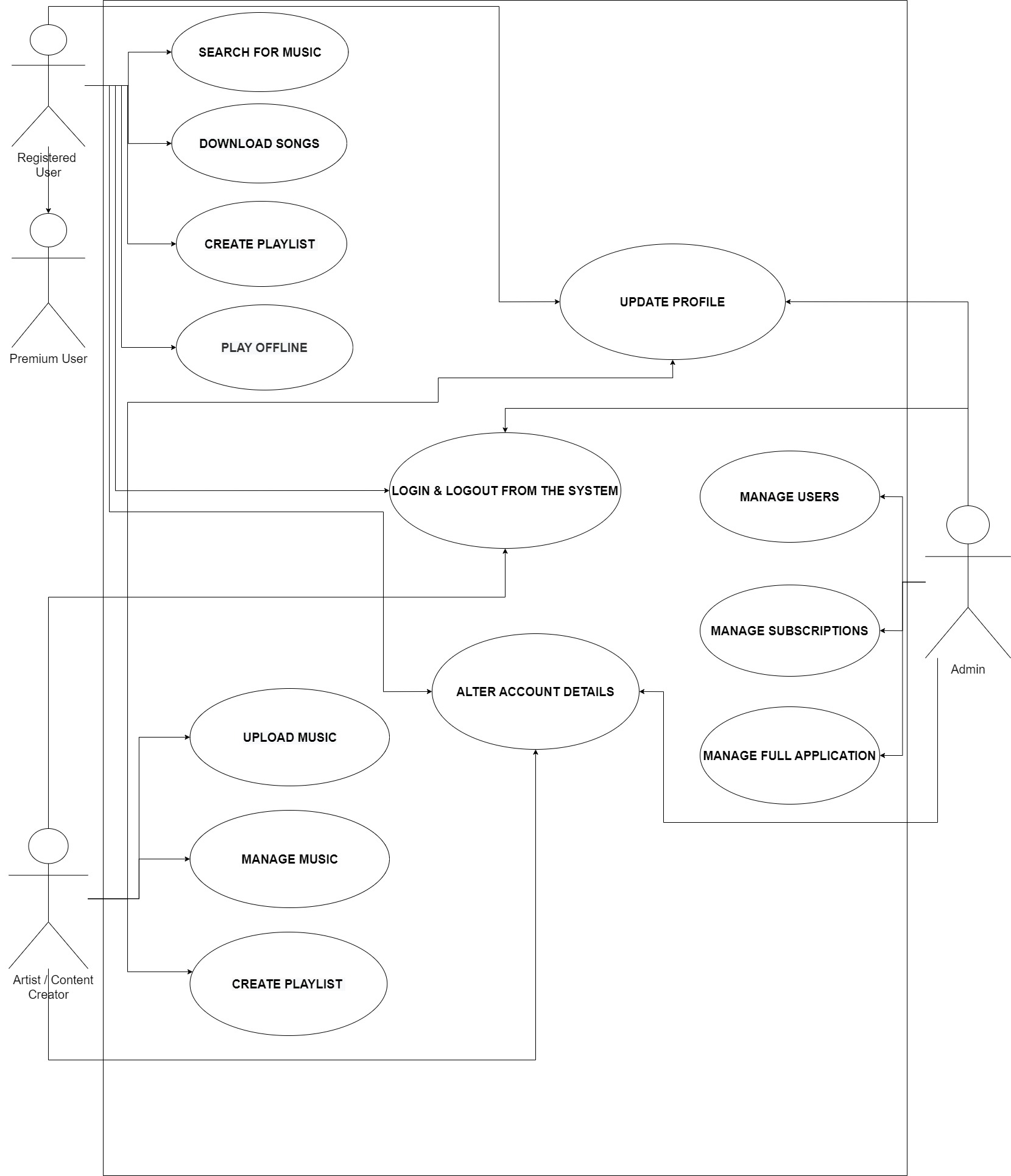
**DATA DICTIONARY:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Key** | **Name** | **Data Type** | **Length** | **Nullable** |
| **USER** | | | | |
| PRIMARY | U\_ID | INTEGER | 10 | NO |
|  | U\_NAME | VARCHAR | 30 | NO |
|  | U\_DATEOFBIRTH | VARCHAR | 30 | NO |
|  | U\_EMAIL | VARCHAR | 20 | NO |
|  | U\_PASSWORD | VARCHAR | 20 | NO |
| **PLAYLIST** | | | | |
| PRIMARY | PL\_ID | INTEGER | 10 | NO |
|  | PL\_TITLE | VARCHAR | 30 | NO |
|  | PL\_CREATIONDATE | VARCHAR | 30 | NO |
|  | PL\_CATEGORY | VARCHAR | 30 | NO |
| FOREIGN | U\_ID | INTEGER | 10 | NO |
| **SONG** | | | | |
| PRIMARY | S\_ID | INTEGER | 10 | NO |
|  | S\_TITLE | VARCHAR | 20 | NO |
|  | S\_GENRE | VARCHAR | 30 | NO |
|  | S\_RELEASEDATE | VARCHAR | 20 | NO |
|  | S\_DURATION | VARCHAR | 50 | NO |
| FOREIGN | AL\_ID | INTEGER | 10 | NO |
| **ARTIST** | | | | |
| PRIMARY | A\_ID | INTEGER | 10 | NO |
|  | A\_NAME | VARCHAR | 20 | NO |
|  | A\_GENRE | VARCHAR | 20 | NO |
|  | A\_BIOGRAPHY | VARCHAR | 50 | NO |
| **ALBUM** | | | | |
| PRIMARY | AL\_ID | INTEGER | 10 | NO |
|  | AL\_NAME | VARCHAR | 20 | NO |
|  | AL\_GENRE | VARCHAR | 50 | YES |
|  | AL\_CATEGORY | VARCHAR | 20 | NO |
|  | AL\_RELEASEDATE | VARCHAR | 20 | NO |
| FOREIGN | A\_ID | INTEGER | 10 | NO |
| **LOGIN** | | | | |
| PRIMARY | L\_ID | INTEGER | 10 | NO |
|  | L\_USERNAME | VARCHAR | 30 | NO |
|  | L\_PASSWORD | VARCHAR | 30 | NO |
|  | L\_ROLE | VARCHAR | 30 | NO |
| FOREIGN | U\_ID | INTEGER | 10 | NO |
| **PREMIUM** | | | | |
| FOREIGN | PAYMENT\_ID | INTEGER | 10 | NO |
|  | P\_AMOUNT | INTEGER | 10 | NO |
|  | DATE | VARCHAR | 30 | NO |
|  | PROMOCODE | VARCHAR | 30 | YES |
| **PAYMENT** | | | | |
| PRIMARY | PAYMENT\_ID | INTEGER | 10 | NO |
|  | P\_PAYMENT\_TYPE | VARCHAR | 30 | NO |
| **SONGS\_IN\_PLAYLIST** | | | | |
| FOREIGN | S\_ID | INTEGER | 10 | NO |
| FOREIGN | PL\_ID | INTEGER | 10 | NO |

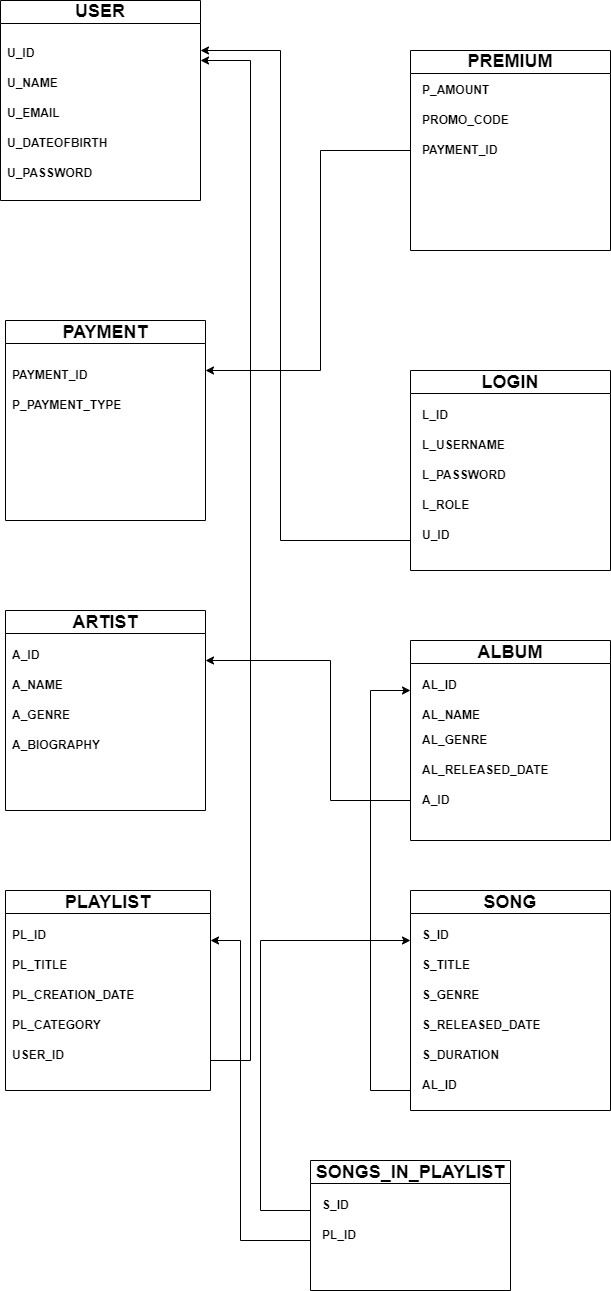
**REQUIREMENTS DESIGN SPECIFICATION**

* + - Use case diagram.
    - Class diagram
    - E-R diagram and Data dictionary
    - User Interface design (optional)

**USE CASE DIAGRAM**



**Class diagram**



**E-R diagram and Data dictionary**

