

**Richard Jones**

A Level Computer Science Coursework

Music Centre Program

Centre Number – 52423

Candidate Number - 2389

Module - H446/03 or 04

Draft

Authors : Richard Jones

Date : 7 February 2017

Version : V0.1

E-mail : richardj99@sky.com

Contents

[1 Introduction 3](#_Toc478643934)

[2 Analysis 4](#_Toc478643935)

[2.1 Suitability for Solubility by computational Methods 4](#_Toc478643936)

[2.1.1 Description of Relevant Features of problem 4](#_Toc478643937)

[2.1.2 Why is this problem Suitable for solution by computational Methods? 4](#_Toc478643938)

[2.2 Stakeholder Management 4](#_Toc478643939)

[2.2.1 Who are the stakeholders and how were they chosen? 4](#_Toc478643940)

[2.2.2 How will they make use of the proposed solution? 4](#_Toc478643941)

[2.2.3 Why will the approach taken be suitable for stakeholder needs 5](#_Toc478643942)

[2.2.4 Research into Alternative Solutions 5](#_Toc478643943)

[2.2.5 Minimum Essential features required 7](#_Toc478643944)

[2.2.6 Limitation of Minimum Features 8](#_Toc478643945)

[2.2.7 Identification of proposed solution with reference to Research 8](#_Toc478643946)

[2.2.8 Hardware and software requirements for solution 9](#_Toc478643947)

[3 Design 10](#_Toc478643948)

[3.1 Breakdown of Problem 10](#_Toc478643949)

[3.1.1 Problem Elements suitable for computational solutions 10](#_Toc478643950)

[3.1.2 Justifying the process 10](#_Toc478643951)

[3.2 Detailed Structure of Solution to be developed 10](#_Toc478643952)

[3.2.1 User Interface Design 10](#_Toc478643953)

[3.2.2 Database Design 17](#_Toc478643954)

[3.2.3 Algorithms required in solution 18](#_Toc478643955)

[3.3 Key Features of solution and their justification 18](#_Toc478643956)

[3.3.1 Structure of solution 18](#_Toc478643957)

[3.3.2 Features of solution 18](#_Toc478643958)

[3.3.3 Test Data to be used 18](#_Toc478643959)

[4 System Development 19](#_Toc478643960)

[4.1 Development iterations 19](#_Toc478643961)

[4.1.1 Order of development 19](#_Toc478643962)

[4.1.2 Prototype Versions 19](#_Toc478643963)

[4.1.3 Modular Structure 19](#_Toc478643964)

[4.1.4 Annotation of Code and Variable naming 19](#_Toc478643965)

[4.1.5 Validation of Key elements of solution & Evidence of stage review 19](#_Toc478643966)

[5 Testing 20](#_Toc478643967)

[5.1 Breakdown of testing Activity and any system changes resulting from it 20](#_Toc478643968)

[5.1.1 User Log on Process 20](#_Toc478643969)

[5.1.2 System Parameters process 20](#_Toc478643970)

[5.1.3 Data Import process 20](#_Toc478643971)

[5.1.4 Song Play 20](#_Toc478643972)

[5.1.5 Artist Play 20](#_Toc478643973)

[5.1.6 Album play 20](#_Toc478643974)

[5.1.7 Transport Button functionality 20](#_Toc478643975)

[5.1.8 Search function 20](#_Toc478643976)

[5.1.9 Queue function 20](#_Toc478643977)

[5.1.10 Playlist creation 20](#_Toc478643978)

[5.1.11 Playlist use 20](#_Toc478643979)

[6 Evaluation 21](#_Toc478643980)

# Introduction

# Analysis

## Suitability for Solubility by computational Methods

### Description of Relevant Features of problem

Users require many features from a modern ‘Music Player’ program. First and foremost, it needs to play the user’s music files and allow the user to control its playback (such as functions allowing it to pause and resume).

Users also want a program that can provide a logical view (or library) of their files, displaying information about their songs (such as song names and track numbers). The typical User normally would want to traverse this library, scrutinising by albums and artists as well as querying the library for songs, albums and artists.

The ability to create playlists is also highly sought after. These playlists should be nameable and will allow the users to add songs to them. They also want ‘Smart Playlists’ that collect data on their activity and create playlists in reflection of that (such as most played songs).

These features are standard for many other music player programs. However, I aim to produce a more conservative player that contains only the basic features and by doing so, create a practical music player without significantly affecting the performance of the computer.

### Why is this problem Suitable for solution by computational Methods?

This problem has already been solved by many other companies and has been implemented by many companies in commercial products. This fact shows that the program must be solvable by computational methods.

The idea of creating a library of music files and allowing their playback will require many different methods, many of which are provided in all programming languages. The constructs of iteration, modules and object-oriented programming can be used to produce a suitable and efficient solution to this problem. Furthermore, databases can be used to store and normalise all the data in the program, ensuring the validity of its data.

## Stakeholder Management

### Who are the stakeholders and how were they chosen?

There could be many potential stakeholders for this solution as most of the population have a mutual enjoyment of music and would be interested in the media player program described.

To collate as wide a target market as possible, I have selected several different people to be my stakeholders. These people will originate from my tutor group, my school’s staff and my family. They will express their wants and needs out of a solution to this problem and will be consulted throughout the design, development and testing process of this program.

### How will they make use of the proposed solution?

Several studies (and my own personal experience) have shown that people tend to listen to music while they are working and relaxing, either at home or at a workplace. I aim for my solution to be used in these situations.

The solution will be in the form of an application that people can keep open on their desktop, allowing them to concentrate on work or relax while using the solution.

Furthermore, I have conducted several interviews with the stakeholders to create a more specific picture of what my solution needs to satisfy their needs. Here is the survey that I gave to all of them:

|  |
| --- |
| **Name:** |
| **Age:** |
| **Vocation:** |
| **Which program do you currently use to play music?** |
| **What, in your opinion, are the unique strengths of this solution?** |
| **What, in your opinion, are the limitations of this solution?** |
| **What feature would you personally want most out of a solution tailored to your needs?** |

The results from this survey were then collated to allow the data to be viewed and concluded more easily. Here are the results:

* **Which program do you currently use to play music?**
  + Windows Media Player **– 2**
  + ITunes – **3**
  + Other – **1**
* **What, in your opinion, are the unique strengths of this solution?**
  + Wide support for files – **2**
  + Online store – **1**
  + Video Playback – **1**
  + Smart playlists/Genius Playlists - 2
* **What, in your opinion, are the** **limitations of this solution?**
  + Slows down machine – **5**
  + Too many pointless features – **1**
* **What feature would you personally want most out of a solution tailored to your needs?**
  + Lightweight solution/won’t interfere with machine performance – **3**
  + Playlists, both automatically created – **3**

Conclusion

Some conclusions can be made from the data gathered in this interview. First, my selected stakeholders show a great care in how the solution will affect the performance of their machine when it is running.

### Why will the approach taken be suitable for stakeholder needs

As previously mentioned, my solution is going to be a lightweight, yet useful, attempt at the ‘music player’ problem. This will be suitable as many of the stakeholders mentioned how their preferred music players produced issues with the performance of their PC.

Furthermore, my stakeholder interviews have showed only a small number of users are interested in video playback. This could be because of the advancements in video streaming products such as Netflix and Amazon Prime Instant Video. These solutions do not require a player to play their files, it is all included in one system. However, my will to focus on music playback perfectly suits the stakeholders’ needs.

### Research into Alternative Solutions

The main competitors for this solution are Microsoft’s Windows Media Player and Apple’s ITunes.

|  |  |
| --- | --- |
| Product Name: | Windows Media Player |
| Features: | * Plays music (with music navigation buttons and progress bar). * Oorganises local media files into libraries * Allows video and photo playback * Allows connection to Network Accessible Devices * Provides plugins for cosmetic features such as player themes and visualizations for music. |
| Supported File Formats: | * Windows Media Formats (.asf, .wma, .wmv, .wm) * Windows Media Metafiles (.asx, .wax, .wvx, .wmx) * Windows Media Metafiles (.wpl) * Microsoft Digital Video Recording (.dvr-ms) * Windows Media Download Package (.wmd) * Audio Visual Interleave (.avi) * Moving Pictures Experts Group (.mpg, .mpeg, .m1v, .mp2, .mp3, .mpa, .mpe, .m3u) * Musical Instrument Digital Interface (.mid, .midi, .rmi) * Audio Interchange File Format (.aif, .aifc, .aiff) * Sun Microsystems and NeXT (.au, .snd) * Audio for Windows (.wav) * CD Audio Track (.cda) * Indeo Video Technology (.ivf) * Windows Media Player Skins (.wmz, .wms) * QuickTime Movie file (.mov) * MP4 Audio file (.m4a) * MP4 Video file (.mp4, .m4v, .mp4v, .3g2, .3gp2, .3gp, .3gpp) * Windows audio file (.aac, .adt, .adts) * MPEG-2 TS Video file (.m2ts) |
| Support: | 24/7 Help Centre provided by Microsoft |

|  |  |
| --- | --- |
| Product Name: | ITunes |
| Features: | Plays Music, organises local media files into libraries, allows video and photo playback, contains a store where users can buy music and video. |
| Supported Music File Formats: | .mp3, .aiff, .wav, .mpeg, .aac, .m4a, .mov, .aac, .ogg |
| Support: | 24/7 Help Centre provided by Apple |

Conclusion

While both of these solutions have strengths and weaknesses in areas, both provide exemplar solutions to the problem in discussion. The most prominent being Windows Media Player’s unprecedented selection of supported file types and ITune’s online store, providing original features to the problem in question.

However, through research, I discovered that both solutions have had many reports of slowing down the performance of machines, particularly ITunes. While the additional features that they provide are appreciated by many people, it seems they come at a cost in the usability of the program. The main point that my stakeholders brought to my attention in the interview is that they require a solution that has little to no impact on their machine. This means that some features of the solution (including some that are in these solutions) must be ignored to ensure its usability.

It is also appropriate to mention that both solutions are professional attempts, meant for a large-scale audience in a largely commercial environment. Because of this, these solutions are likely to have taken several years to perfect, requiring the help of a large team working on the solution at the same time. My solution is meant for a smaller audience and is scheduled to be done in less than a year, meaning that many of the features must be forgotten to meet the target by the required deadline.

It is also unlikely that my solution will be able to provide the 24/7 support, that these two solutions allow, in the immediate future. To address this, I have decided that the program will be open-source. This will have a very small effect on the solution in question, just that the program will have to be well structured with clear variable names to accommodate third-party development.

### Minimum Essential features required

1. Form a normalized database to store information relating to the music.
2. Create a table for the songs.
3. Create a table for the albums.
4. Create a table for the artists.
5. Create a table for the playlists.
6. Create a table that stores information about the program’s users.
7. Provide an input box where users can enter their username.
8. Provide an input box where users can enter their password.
9. Provide a button that users can use to log in to the program.
10. Validate the username and password against the program’s database.
11. Provide access to the main program if validation is successful.
12. Allow permissions to be assigned to each account, giving the specified accounts administrator permissions.
13. Allow administrators to delete playlists and accounts and give other users administrator permissions.
14. Provide a settings pane in the program that allows users to configure the application.
15. Allow users input a directory to import music from.
16. Scan through all files in the directory.
17. Read the tags for all mp3 files.
18. Read the artist data from the tags, writing them in to the artist database.
19. Read the album data from the tags, writing them in to the album database.
20. Read the song data (song name and genre) from the tags, writing them in to the songs database.
21. Provide a view that will show different views depending on a user’s input:
    1. An artist button that will show a view of all the artists in the music library.
    2. An album button that will show a view of all the albums in the music library.
    3. A songs button that will show a view of all the songs in the music library.
22. Allow the user to further scrutinize their view through interaction:
    1. Selecting a field in the album view will display all the songs in the selected album.
    2. Selecting a field in the artist view will display all the albums by the selected artist.
23. Provide a search box where users can manually input queries for songs, artists, albums and playlists
24. Play a selected song when said song is selected in the view.
25. Provide details of the current song playing, including the song name, the artist’s name and the album’s name
26. Allow a song to be paused through the press of a button.
27. Allow a song to be resumed through the press of a button.
28. Provide a progress bar so users can see how much of the song has played/is remaining.
29. Create a queue from the view when a song is played e.g. if a song is selected, all songs below it in the view are added to the queue.
30. Allow the queue to be traversed backwards and forwards through two separate buttons e.g. skip back and skip forward.
31. Provide a playlist pane which allowing users to create and manage playlists:
    1. Create playlists, specific to that user only, and add songs to them.
    2. Allow playlists to be edited e.g. adding or removing songs.
    3. Allow playlist to be deleted by the owner or an administrator.
32. Automatically create a playlist for all users, displaying the most played songs.
33. Place all information about playlists into a normalized database.

### Limitation of Minimum Features

There are several limitations to my proposed solution:

Firstly, my solution will only cater for music playback and not photo or video. This decision was made to reduce the strain that the solution had over the machine it was running on. Better suiting the stakeholder’s requirements.

Secondly, my solution will only support .mp3 files. MP3 files are the most used audio files to date, but there are many other widely used formats which could need support. However, many of these files do not contain the same tags as mp3 (ID3), meaning that my program will not be able to read them into the database. This decision was made to keep the solution as ‘cut down’ as possible as well as reducing the time of development to fit in with the deadline. Better suiting the stakeholder’s needs.

### Identification of proposed solution with reference to Research

Through the research into similar solutions, I discovered that both solutions have had many reports of slowing down the performance of machines, particularly ITunes. While the additional features that they provide are appreciated by many people, it seems they come at a cost in the usability of the program. The main point that my stakeholders brought to my attention in the interview is that they require a solution that has little to no impact on their machine. This means that some features of the solution (including some that are in these solutions) must be ignored to ensure its usability.

It is also appropriate to mention that both solutions are professional attempts, meant for a large-scale audience in a largely commercial environment. Because of this, these solutions are likely to have taken several years to perfect, requiring the help of a large team working on the solution at the same time. My solution is meant for a smaller audience and is scheduled to be done in less than a year, with minimal help from outside sources. This means that many of the features must be forgotten to meet the target by the required deadline.

### Hardware and software requirements for solution

This solution will be programmed in Python 3.4. Python was chosen because it is an object-oriented language, allowing the solution to use models, methods and classes. These constructs will aid the solution’s development because it will allow the problem to be split up into more solvable sections, reducing space and producing a more efficient solution.

Python was also chosen due to its large collection of libraries. Libraries will be used in this solution to read the ID3 tags and allow the playback of songs. The use of these libraries will reduce the amount of time that the solution will be developed, as well as decrease the amount of code in the core program.

# Design

## Breakdown of Problem

### Problem Elements suitable for computational solutions

I aim to decompose the problem down into smaller sub-problems in the hopes that this will make the problem easier to solve. These are the steps that will be taken:

1. First, the application’s User Interface will be designed. The UI will comprise of several windows each serving different purposes. The windows will be the login screen, the create account screen, the media player screen and the settings window. Later in this section, the initial design of these screens will be established by the stakeholders. The UI will be developed in PyQt Designer.
2. After the UI has been designed, the database will be established. This database will hold all the data used by the program, including data relating to the music files (songs, artists, albums and playlists) and data related to the users (usernames and passwords).
3. Once the infrastructure for the application has been implemented, the solution will be coded. This process will start with the coding for the login section. This section will accept account information from the user and validate it in the program’s database. If the details are accepted, the user is admitted into the main program.
4. After the login section, the import function will be programmed in the Settings Window. The function will accept a user specified directory and scan it for any mp3 files, inputting the details from the files into the database.
5. Then the view of the Music Player’s library will be coded. Buttons will be implemented to allow the user to view all the tables in the database (artists, songs, albums etc.) and allow the user to traverse the music library (selecting an artist will retrieve the albums by that artist, selecting an album will retrieve the songs by that artist and selecting a song will play it).
6. The music playback functions will then be coded. This will play a song when it is selected from the view. A pause button will also allow a song to be stopped momentarily. A play button will also be provided to resume a paused song.
7. The queue function will then be coded. The function will add all of the songs under a selected song to a queue. Once the selected song has finished playing, the next song in the queue will be retrieved. This will continue until the queue is empty or a new song is selected.
8. After the queue function has been implemented into the program, the skip buttons will be programmed to traverse the queue backwards and forwards.
9. After the song queueing function has been programmed, the search function will be programmed. The search function will accept a string and use it to search the Music Library for any similar fields in the records. All those that match the User’s input will be outputted in the view. The function will also be able to scrutinize what to search for (over artists, albums, songs and playlists).
10. Then the playlist feature will be coded. The feature will allow users to add new playlists into the database as well as access existing ones. A window can then be accessed that will allow the user to add and remove songs from the playlist.

### Justifying the process

## Detailed Structure of Solution to be developed

### User Interface Design

After having a discussion with the stakeholders for this product. I drew up a plan of the interface that everyone could agree on. Slight changes are likely to be made to these in order to improve functionality.

#### **Login Window**

Username:

Password:

Log-In

Create Account

Please Login

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Number** | **Purpose** | **Example Input** |
| txt\_username | 1 | This is a textbox where the user can input their username. | “user1” 🡪 String |
| txt\_pass | 2 | This is a textbox where the user can input their password. | “password1” 🡪 String |
| btn\_login | 3 | This is a button that, once clicked, uses the information that the user has provided and attempts to gain access to the main program. | N/A |
| btn\_crtacc | 4 | This is a button that, once clicked, will take the user to the ‘create account’ window. The user can then proceed to create a new account. | N/A |
| lbl\_username | 5 | This label currently displays ‘Please Login’. However, if any errors are brought up when checking the user’s inputs (e.g. username doesn’t exist in database), a suitable message will be displayed. | N/A |
| lbl\_password | 6 | This label is used to indicate where the user’s password should be entered. | N/A |
| lbl\_info | 7 | This label is used to indicate where the user’s username should be entered. | N/A |

#### **Create Account Window**

Create an Account

Create Account

Password:

Username:

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Number** | **Purpose** | **Example Input** |
| txt\_username | 1 | This object allows the user to input their desired username. | “User1” 🡪 String |
| txt\_password | 2 | This object allows the user to input their desired password. | “Password1” 🡪 String |
| lbl\_username | 3 | This label indicates where the user should enter their username. | N/A |
| lbl\_password | 4 | This label indicates where the user should enter their password. | N/A |
| lbl\_info | 5 | This label initially displays “Create an Account”. However, if any errors are raised when validating the user’s inputs, the label will show a suitable message (e.g. Username has already been taken). | N/A |
| btn\_crtacc | 6 | This button will activate the procedure that takes the user’s inputs and creates an account using them (provided the inputs are successfully validated) | N/A |

#### **Media Player Window (Main Window)**

Songs View

Artists

Albums

Songs

Playlists

Settings

Playlist Manager



Now Playing

Search

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Number** | **Purpose** | **Example Input** |
| Btn\_back | 1 | When this button is activated, it will play the song at the top of the ‘previously playing’ stack. | N/A |
| Btn\_forward | 2 | When this button is activated, it will play the song at the front of the “next song” queue. | N/A |
| Btn\_pause | 3 | When this button is activated, the song that is currently playing will be paused. | N/A |
| Btn\_play | 4 | When this button is activated, any song that has been paused will be resumed. | N/A |
| Lbl\_nowplaying | 5 | Initially, this label will display: “Now Playing:”. However, when a song begins playback, the name of the song will be added to the end of the label. | N/A |
| Btn\_exit | 6 | When this button is activated. The entire program will shut down. | N/A |
| Btn\_search | 7 | When this button is activated, The input from the drop-down menu and the search box will be collected and the music library will then be searched (using these inputs). | N/A |
| Drp\_search | 8 | This drop-down menu allows the user to specify whether they want to search for Albums, Songs, Artists or Playlists | N/A |
| txt\_search | 9 | This object allows the users to input the term that they want to search for in the Music Library. | “ArtistName” 🡪String |
| Lbl\_album | 10 | This label will show which album (if any) is being viewed in the Song View | N/A |
| Lbl\_artist | 11 | This label will show which artist (if any) is being viewed in the Song View | N/A |
| Lbl\_nowviewing | 12 | This label is to help the user understand what the lbl\_artist and lbl\_album objects are denoting | N/A |
| Btn\_albums | 13 | When this button is activated, the song view (tbl\_songs) will show all the albums in the music library | N/A |
| Btn\_artists | 14 | When this button is activated, the song view (tbl\_songs) will show all the artists in the music library. | N/A |
| Btn\_playlists | 15 | When this button is activated, the song view (tbl\_songs) will show all the playlists in the music library. | N/A |
| Btn\_plylstmngr | 16 | When this button is activated, the ‘Playlist Manager’ window will be opened on top of the ‘Main’ Window. | N/A |
| Btn\_settings | 17 | When this button is activated, the ‘Settings’ Window will be opened on top of the ‘Main’ Window | N/A |
| Tbl\_songs | 18 | This table will show all the fields in the libraries database. At a basic level, it will show all of artists, artists, songs and playlists (depending on which button the user selects). However, it will also display albums belonging to a certain artist, songs belonging to a certain album and songs belonging to a certain playlist | N/A |

#### **Settings Window**

Import

Directory:

Users Table

Delete User

Toggle Admin.

Logout

Exit

**User: User1**

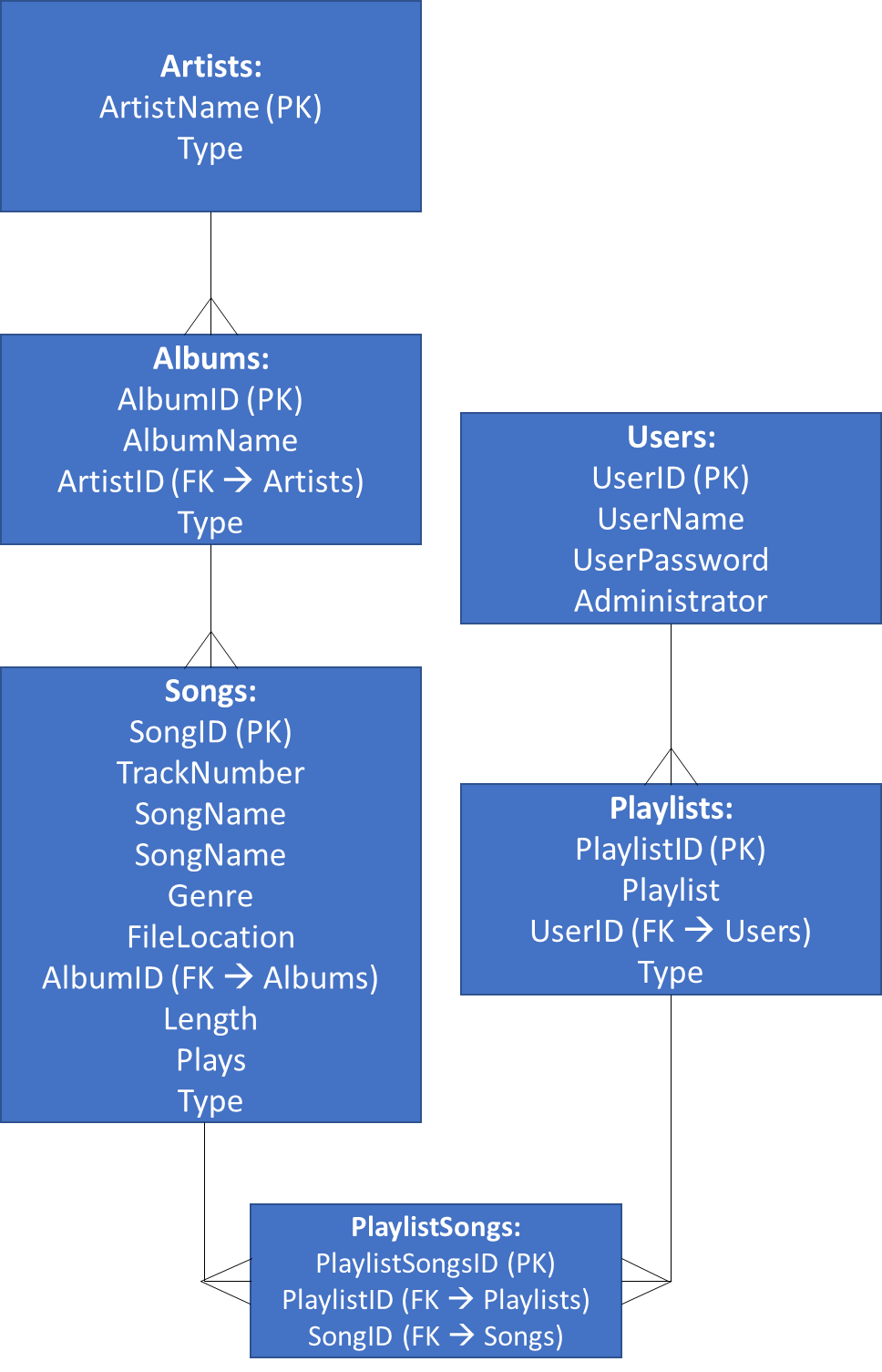
|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Number** | **Purpose** | **Example Input** |
| Btn\_exit | 1 | When this button is activated, this window will be closed. | N/A |
| Btn\_logout | 2 | When this button is activated, the user will be taken back to the login screen. Any variables that identify the user (e.g. UserID and UserName) will be set to a null value. | N/A |
| Btn\_admin | 3 | When this button is activated, the selected user (the field selected in tbl\_users) will be made an admin or, if they are already an admin, will be made an ordinary user. | N/A |
| Btn\_delUser | 4 | When this button is activated, the selected user (the field selected in tbl\_users) will be deleted from the program and the database. | N/A |
| Lbl\_user | 5 | This object has a null value of “User:”. However, when a user is selected from tbl\_users, the object will also show the name of the user. | N/A |
| Tbl\_users | 6 | This table will show all of the users (from the users table in the database), excluding the user who is currently logged in. | N/A |
| Btn\_import | 7 | When this button is activated, the user’s input (their selected directory) from txt\_dir is taken. The music in this directory is then imported into the database. | N/A |
| Lbl\_dir | 8 | This object will indicate where the user should enter their directory. | N/A |
| Txt\_dir | 9 | This object will allow the user to input the directory from where they want their music to be imported from. | “C:/Users/User/Music/” 🡪 String |

#### **Playlist Manager Window**

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Number** | **Purpose** | **Example Input** |
| Btn\_close | 1 | When this button is activated, the window is closed, taking the user back to the Main Window |  |
| Btn\_newplylst | 2 | When this button is activated, a dialog box will open. Here the user can input the new playlist’s name. |  |
| Btn\_plylstreset | 3 | When this button is activated, th |  |
| Tbl\_playlists | 4 | This table will display all of the information relating to playlist. This includes the names of the playlists and the songs that each playlist contain. |  |
| Tbl\_songs | 5 | This table will show all of the songs that are currently in the music library. |  |

### Database Design

In order to maintain complete and consistent efficiency in my program, I will need to have a normalised database, preferrably in Third Normal Form.

In order to do this, I have decided to isolate the three most independent fields to describe songs (The artist, album and title)

### Algorithms required in solution

Import Function

This function will scan all of a specified directory. It will identify any mp3 files in this directory and obtain the tags from them. These tags will then be added into the

## Key Features of solution and their justification

### Structure of solution

Defined in detail the structure of the solution to be developed.

Described the solution fully using appropriate and accurate algorithms justifying how these algorithms form a complete solution to the problem.

### Features of solution

Described, justifying choices made, the usability features to be included in the solution.

Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) justifying and explaining any necessary validation.

### Test Data to be used

3,600 songs as mp3s. These will be used as test data.

Identified and justified the test data to be used during the iterative development of the solution.

Identified and justified any further data to be used in the post development phase.

# System Development

## Development iterations

### Order of development

1. Login Screen 🡪 started with single user not in database
2. Import Function 🡪 MP3 validation, resets database every time import is clicked. later release can import the changes only
3. Library Tables
4. Player functions

Provided evidence of each stage of the iterative development process for a coded solution relating this to the breakdown of the problem from the analysis stage and explaining what they did and justifying why.

### Prototype Versions

Provided evidence of prototype versions of their solution for each stage of the process.

### Modular Structure

The solution will be well structured and modular in nature.

### Annotation of Code and Variable naming

Code will be annotated to aid future maintenance of the system.

All variables and structures will be appropriately named.

### Validation of Key elements of solution & Evidence of stage review

There will be evidence of validation for all key elements of the solution.

The development will show review at all key

# Testing

## Breakdown of testing Activity and any system changes resulting from it

### User Log on Process

### System Parameters process

### Data Import process

### Song Play

### Artist Play

### Album play

### Transport Button functionality

### Search function

### Queue function

### Playlist creation

### Playlist use

# Evaluation