ECSE 321 – Introduction to Software Engineering – Group 1

Final Deliverables

Mission 4: Black box test plan and Implementation

Samuel Cormier-Iijima (260174995), Simon Foucher (260223197), Stefanos Koskinas (260211145), Bertin LeBlanc (260191026), Amin Mirzaee (260209556)
4/2/2009

Contents

1 Use Case Model

1.1 Use Case Diagrams

The following diagrams illustrate some of the use cases for the music player as well as some of the interactions between them and the actors in the use case model. These diagrams are meant to provide a graphical view of the use case analysis that follows in the next section.

Figure 1 illustrates some of the most important use cases that refer to the user actor. These actions/functions will very often be used when operating the music player.

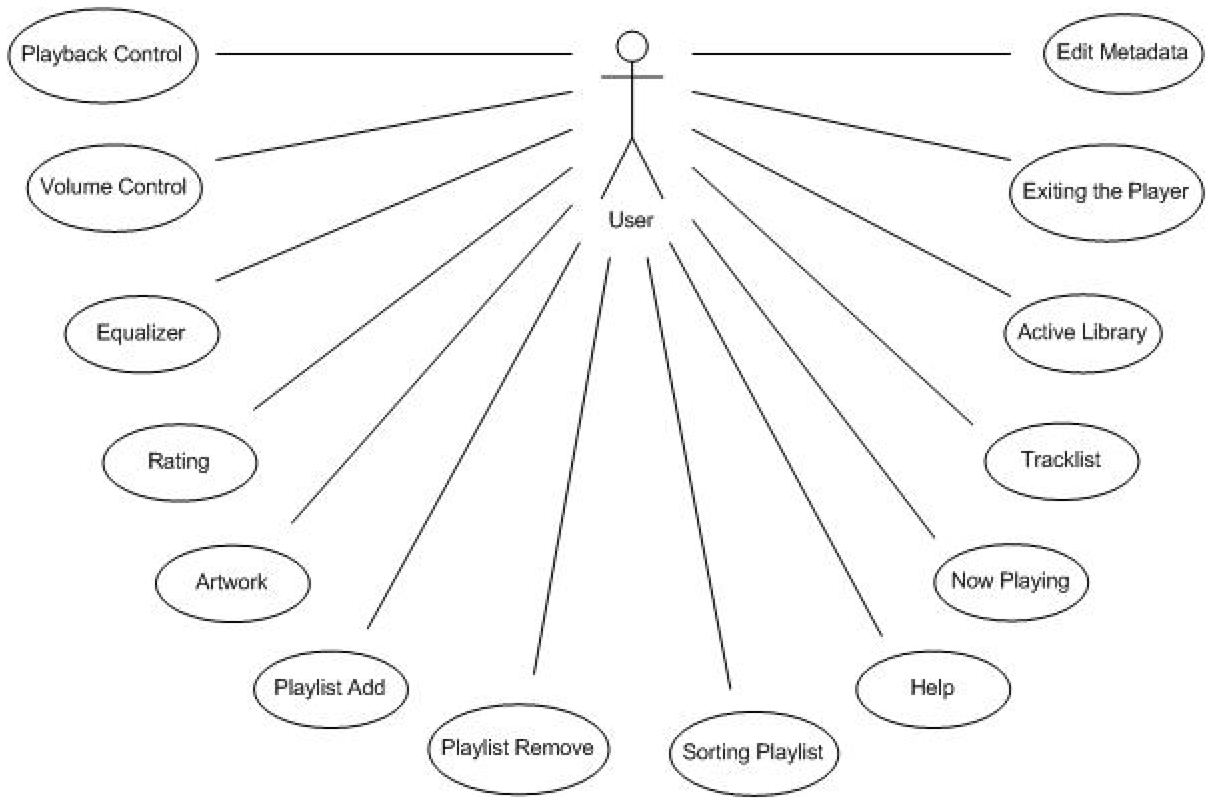

Figure 1: Music Player User and Use Cases

Figure 2 below illustrates some of the interactions between different functions of the system. This diagram also points out the <<include>>, <<extend>> and inheritance relationships of these particular functions.

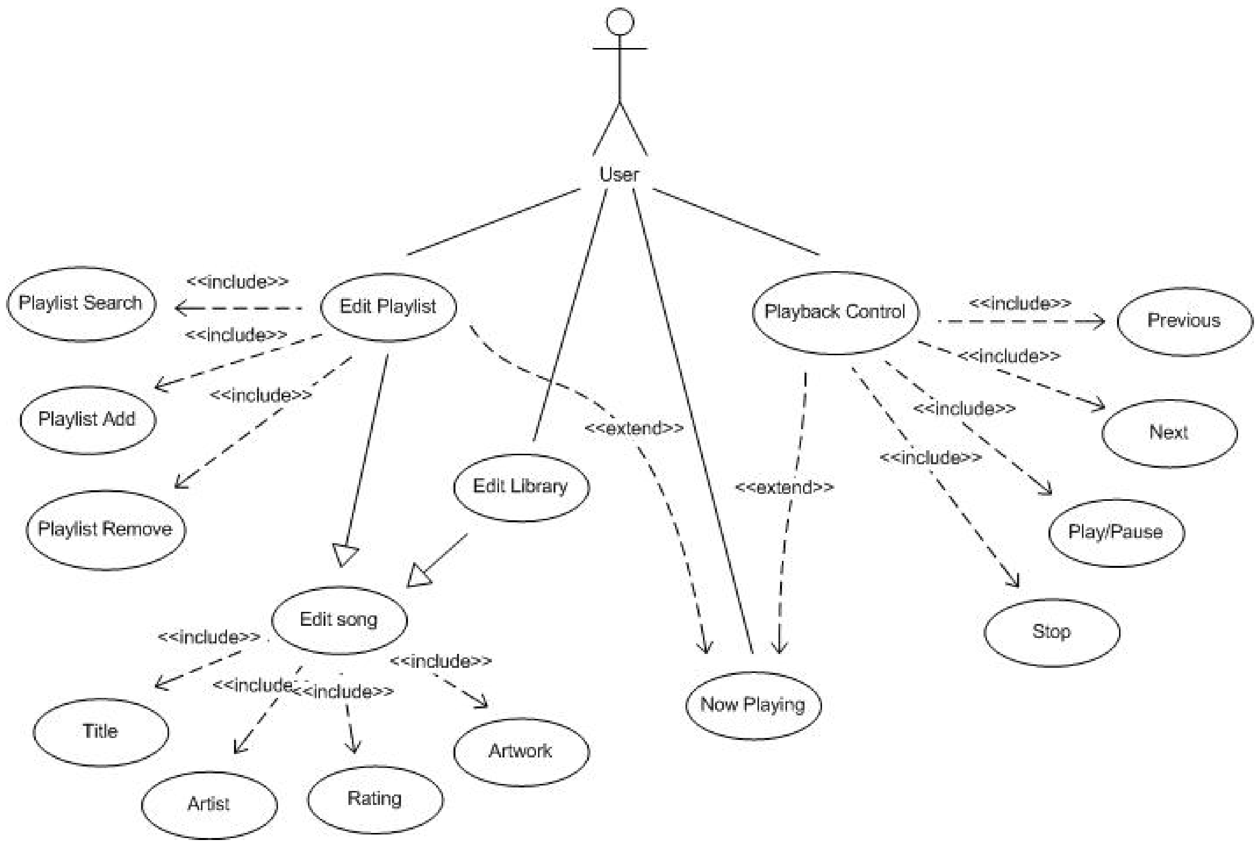

Figure 2: Interactions Between User and System Functions

For this illustrative example, four main use cases are shown for the user: Playback Control, Now Playing, Edit Library and Edit Playlist. Playback Control includes the individual actions represented by buttons on the player's interface: play/pause, stop, next and previous. Similarly, Edit Playlist also includes more specific edit actions, such as add, remove and search. The Edit song function inherits a major portion of characteristics, the difference being that in this case the editing aims at the song metadata. The Edit song function is included in both the Edit Playlist and Edit Library, as they both share similar features. Finally, the Edit Playlist and Playback Control functions extend the Now Playing function.

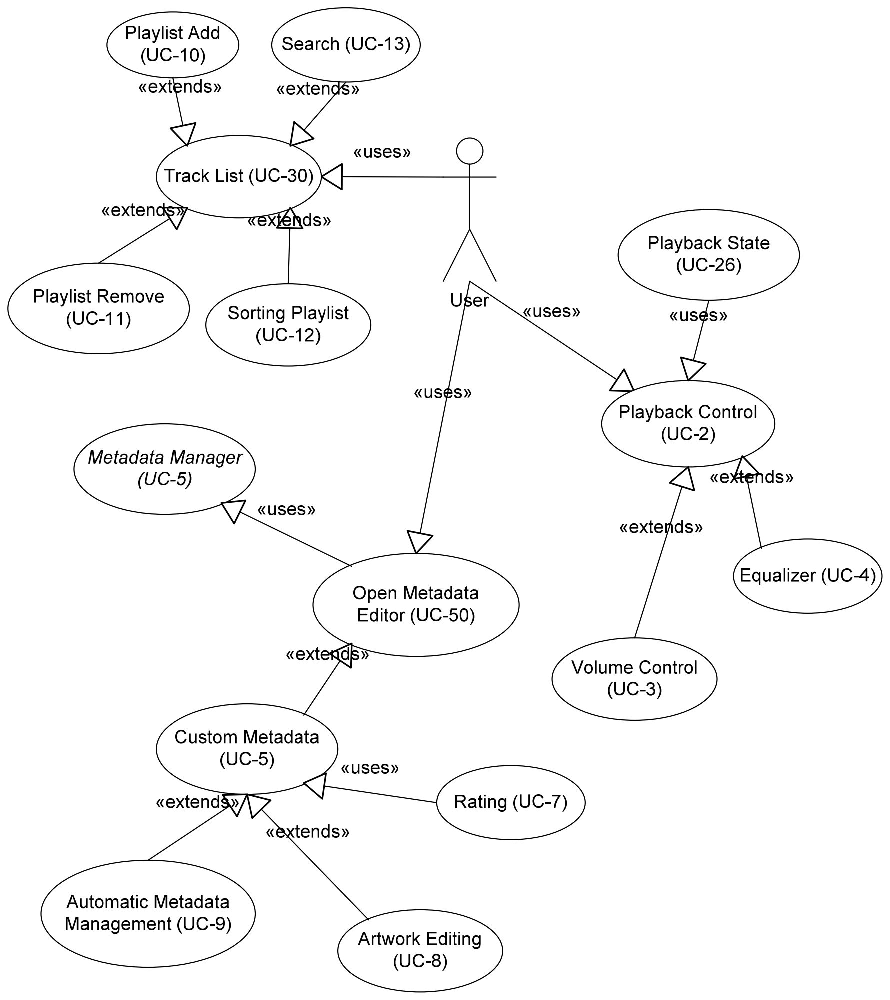

Figure 3: Use Case Diagram Traceable to Use Case Identifiers

1.2 List of Actors

(Listed in order of frequency of interaction with the system)

- **User**: The user represents any person who will be using the player. When the user uses the player, it is already installed and configured on a computer
- **Installer**: This actor represents the person who will be installing the software on a PC. The installer has access to the music player's "setup.exe" or .deb file and a PC running Windows 32 bits or Linux. The installer's primary objective is to install the software.
- **Hacker**: The hacker has the music player installed on his machine and wishes to steal playlists from other Remote Users without their consent, or regardless of the configuration of their players.
- **Lawyer**: The lawyer wants to see if the player conforms to copyrights regulations and if not, sue the software developers for everything they are worth.

1.3 List of Use Cases

Identifier	Use Case	Description
UC-1	Audio Format	The **user** wants to play files encoded in one of the major audio formats. He installs the appropriate audio codecs. Afterwards, the player will recognize the audio files.
UC-2	Playback Control	The **user** is listening to a song, and he wants to stop, pause, or seek. He presses on the appropriate buttons on the GUI. The song's playback status is updated accordingly.
UC-3	Volume Control	The **user** is listening to a song and wishes to modify the volume of the player's audio output. He slides the volume slide bar or presses the mute button. The volume level by the player is updates accordingly.
UC-4	Equalizer	The **user** wishes to modify the gain of specific audio frequencies outputted by the player. He clicks on the equlalizer button in the main window and slides the equalizer scales representing the band he wishes to amplify.
UC-5	Metadata Manager	The **user** wishes to create, update or delete metadata associated with an audio file (i.e. song title, album, cover, etc…). The user clicks the "edit metadata" menu and edits the songs he wishes.
UC-6	Custom Metadata	The **user** wants to create his own personal fields of metadata and name them. He clicks on the appropriate menu item and a dialogue opens.
UC-7	Rating	The **user** wants to save a ratings he sees fit for particular songs. He chooses "Edit Rating" in the menu function.
UC-8	Artwork Editing	The **user** wants to change the artwork associated with an audio file. He selects the appropriate menu button and updates the image file.
UC-9	Automatic Metadata Management	The **user** wants the player to automatically manage the metadata associated with the songs. He enters that choice in the configuration option of the menu. The player will extract the metadata it can from the audio file and update it on the player.
UC-10	Playlist Add	The **user** wishes to add songs to a playlist. He clicks the song from the main view and drags it to the playlist.
UC-11	Playlist Remove	The **user** wishes to delete a song from a playlist. He selects the playlist such that it is visible in the main window. He selects the undesired song and clicks on it to highlight it, presses "delete" then "save playlist".
UC-12	Sorting Playlist	The **user** wants to change the order in which the songs are played within a playlist. He clicks and drags them up and down. If he wants to make those changes permanent, he clicks on the "save playlist" button

UC-13	Search	The **user** wants to search for a particular song within a playlist. He enters the song name in the search field and presses "search" button. The playlist containing the song will appear in the main view with the song highlighted.
UC-14	Import Song	The **user** can import a file/directory on the local hard drive by opening the "import window" from the menu. Importing an audio file will cause that file to be added to the library. Importing a directory will cause all audio files contained recursively within it to be added to the library.
UC-15	Default Library	The **user** wants to have a default import library for the songs imported. He enters the configuration panel via the menu and chooses default library.
UC-16	Network Song Access	The **user** winches to select which network location will be browsed for songs. This is done via the configuration panel, accessed from the menu.
UC-17	Remote Song Import	Once viewing the "Network songs", the **user** can select specific files and click on the "import" button to import those files to his local song library.
UC-18	Share Playlists	In the configuration view (accessed from the menu), the **user** can select whether or not he wishes to share particular songs/playlists
UC-19	Auto-Sync	The **user** wants one of his playlist to synchronize automatically with a remote repository. He selects the option in playlist configurations.
UC-20	Help	The **user** needs help with some of the software's features. He selects the help entry in the menu and a help window pops up. He can then scroll the document and use search function.
UC-21	QA	The **user** wants to listen to undistorted music and not have to deal with signal analysis.
UC-22	Consistent Gain	The **user** wants to hear songs at a stable level without having to continuously readjust volume and equalizer
UC-23	Volume Display	The **user** wants to be able to see what the volume is set to within the main view.
UC-24	Equalizer Display	The **user** wants to be able to see the current frequency gain settings before updating them.
UC-25	Now Playing	The **user** wants to get intuitive information the track playing. He selects the "now playing" playlist and sees the song being played in highlighted.
UC-26	Playback State	The **user** can get information on the current playback state by looking at the "time left" at the bottom of the player. If the song is playing, the time remaining is shrinking.

UC-27	Active Track	The **user** can see the Title, Album and Artist associated with the current track playing by looking at the top of the player
UC-28	Artwork Viewing	The **user** can see the Artwork associated with the file currently playing by looking at the top right corner of the player, where it is displayed.
UC-29	Meta Sorting	The **user** wants to be able to brows his playlist when sorted based on a metadata item. He clicks sort by "meta tag" and the list of songs within the visible playlist are sorted by the meta Tag selected.
UC-30	Track List	The **user** can see all the tracks within a playlist by selecting that playlist. The list of tracks will appear in the main vies and he will be able to navigate using a standard sidebar.
UC-31	Aggregate Playlist Information	The **user** can learn the total number of songs within a playlist as well as its total duration by looking at the bottom of the player once the playlist is within the main window.
UC-32	Current Order	The **user** can temporarily change the order of songs within a playlist. To view the current order, he can click on "now playing" and will see a clone of the playlist he originally selected to play.
UC-33	Active Library	The **user** can see all the songs within the active library by clicking on "now playing". This content will be displayed within the main window.
UC-34	Location Information	The **user** can see in the main window weather the music files present on the playlist are located on his local machine on at a remote location
UC-35	Song Statistics	The **user** can get playback information on any songs present within his library by selecting the view entire library tab and looking at the appropriate column
UC-36	Remote Locations	The **user** can see a list of available remote locations from which we may extract songs. This list is accessible from a menu function.
UC-37	Remote Files Available	The **user** can click on a remote location which is available, and a list of available songs will appear within the main window.
UC-38	Remote Metadata Tags	When viewing a song playlist from a remote location, the **user** will also see all the available metadata in the main view.
UC-39	Favorite "Friends"	The **user** can specify a list of known remote network location for future quick access. This is done in the remote locations panel.
UC-40	Documentation	The **user** can consult documentation relevant to the player by selecting the "help" function in the menu.

UC-41	Crawler	The **user** wants the player to keep an updated list of available remote resources on the network without having to continuously refresh and track down location manually.
UC-42	Auto Sync	The **user** wants the player to synchronize automatically with certain pre determined remote locations.
UC-43	Metadata Synchronization	When the **user** changes the metadata associated with a song, the player automatically updates the audio file to reflect that change. Vice versa, whenever a song is refreshed, its metadata is read from the file and displayed on the screen.
UC-44	Auto Playback Statistics	The **user** doesn't have to do anything in particular for playback statistics to get updated. The playback statistics are automatically updated and displayed.
UC-45	Database Loading	When the music player is started by the **user**, the default music library's information is automatically fetched from the DB.
UC-46	Initial Setup	When the **installer** installs the music player on a computer, a database is created to store library information.
UC-47	Initialization of Services	When the **user** loads the music player, the audio filters, network utilities are set up. The player also caches data relevant to the music library to speed up future operations.
UC-48	Background	The **user** does not have to worry about any kind of background operations involving the host environment.
UC-49	Backup Library	The **user** can export the contents of his library by selecting this menu item. The backups created will be importable from any other players.
UC-50	Open Metadata Editor	The **user** can open the metadata editor window by selecting this menu item. Then clicked, a separate window will open (see metadata Window below for more details)
UC-51	Exiting the player	The **user** can close the player using this button. A safe close will ensure that all unsaved data gets saved and that buffers get flushed to the database before terminating the process.
UC-52	Playlist Picker	The **user** can toggle between viewing different playlists and the main song library using this area.
UC-53	Playlist Viewer	The **user** can see all the songs present within a playlist, as well as their respective metadata in this area.
UC-54	Double Click Play	The **user** can play a song simply by double clicking on it in the playlist viewer.
UC-55	Quick Add	The **user** can add a song to a playlist by selecting it in the playlist viewer and dropping it in the desired playlist within the playlist picker.

UC-56	Edit Metadata	The **user** can open the metadata editor window from the menu after highlighting a track. Once open, the **user** will be able to edit the song title, artist, Album, Year, Track number, Genre, Rating, Custom Tags. After editing, pressing the "save" button will record the new data within the database.
UC-57	Intuitive Use	The **user** should be able to figure out how to use the music player intuitively, based on previous experience with other players. Otherwise, full documentation is available to describe its functions.
UC-58	Help Installation	When installing the software, in addition to the music player, the **installer** will install help documentation.
UC-59	Performance	The **user** can load a playlist containing 1000 songs within 1 second on a computer meeting the minimum system's requirements of the player.
UC-60	Reliability	The **user** can terminate the music player process abruptly without any effect on the audio files and data base content on the local/network drives
UC-61	Security	The **hacker** will not be able to access the user's data if the User did not select the share option for that data.
UC-62	Portability	The **Installer** will be able to install the player on a Windows 32 platform using the executable, or on a Linux platform using the .deb package.
UC-63	Legality	The **lawyer** will not be able to sue the software developers on the basis of Canadian computer software regulations.

1.4 Detailed Description of Use Cases

1.4.1 Playback Control (UC-2)

Participating Actor: User

Entry Conditions:

- User has the music player open
- Player is playing a song
- User knows which playback action he wishes to perform

Event Flow:

1. User selects which playback action he wishes to perform (stop, pause or seek the currently playing song) by clicking on the appropriate button on the GUI (in the case of “seek”, the user clicks and drags the slidebar to the desired position in the track, while the time for the current position in the track changes according to the position of the slidebar).
2. In the case of “stop” or “pause”, the song's play ceases and is not resumed until the user presses “play” again. In this case, the user has the ability to change the available options of that song, such as the equalizer options. In the case of “seek”, play resumes as soon as user releases the mouse button and the time of the track is updated accordingly.

Exit Conditions:

- Song is stopped, paused, or seeked to a new time

1.4.2 Volume Control (UC-3)

Participating Actor: User

Entry Conditions:

- User has the music player open
- Player is playing a song

Event Flow:

1. User selects at what volume he wishes his music to play by sliding the volume slidebar to the desired level or by pressing the mute button (this has the effect of completely killing the audio signal) and then releasing the mouse button.

Exit Conditions:

- The audio volume output by the player is updated real-time and the song continues to play at the new volume level

1.4.3 Equalizer (UC-4)

Participating Actor: User

Entry Conditions:

- User has the music player open
- Player is playing a song
- User has the Equalizer window open

Event Flow:

1. Owner selects the desired frequency band gain levels by clicking on, holding then releasing the slidebar representing the frequency band that he wishes to alter. This is repeated for all the frequency bands which the user wishes to alter.
2. The equalizer setting are kept permanently or until they are explicitly changed again.

Exit Conditions:

- Audio output is altered according to the equalizer settings

Exceptional Case:

- If the audio signal does not contain a given frequency band and the user tries to change this band's gain, no change in the output will occur.

1.4.4 Rating (UC-7)

Participating Actor: User

Entry Conditions:

- User has the music player open
- User knows which song's rating he wishes to edit

Event Flow:

1. The user selects the song by clicking on it in the playlist.
2. The user opens “Edit Rating” in the menu and a new window opens, showing the song's current rating.
3. The user adjusts the rating appropriately and then closes the “Edit Rating” window.
4. He repeats this for all songs to which he wishes to apply this action.
5. The song's rating is saved upon exit.

Exit Conditions:

- The desired song's rating is changed and saved

1.4.5 Artwork Editing (UC-8)

Participating Actor: User

Entry Conditions:

- Owner has the music player open

- Owner knows which song's artwork he wishes to change

Event Flow:

1. The user selects the song by clicking on it in the playlist.
2. The user opens “Change Artwork” in the menu and a new window opens, showing the filesystem, allowing the user to select a new image file to use as the song's artwork.
3. The user selects a new image to use as artwork and clicks on “Open”.
4. The new artwork shows up for the current song.

Exit Conditions:

- Song's artwork is updated

Exceptional Case:

- If the file chosen is not an image file or if the image file is corrupted, a message will show up indicating to the user that it is not possible to use this particular file.

1.4.6 Playlist Add (UC-10)

Participating Actor: User

Entry Conditions:

- User has the music player open
- User knows which playlist to add a song to
- User knows which song to add

Event Flow:

1. User selects a song from the main view.
2. User clicks, drags and drops the song into the playlist.

Exit Conditions:

- The song is added to the playlist

1.4.7 Playlist Remove (UC-11)

Participating Actor: User

Entry Conditions:

- User has the music player open
- User knows which playlist to remove a song from
- User knows which song to remove

Event Flow:

1. User selects a playlist by clicking it, so that it shows up in the main view.

2. User selects a song from the playlist by clicking it.
3. User right clicks and selects delete from the options offered.
4. Instead of event 3, the user can simply just hit the delete button
5. Prompt screen will appear asking if the user is sure that he wishes to delete the song.
6. The user then selects yes or no by clicking either option
7. The user saves the playlist.

Exceptional Case:

- The user cannot delete a song from a playlist that is currently playing the song
- Deleting a song from the main library will delete the song from all the playlists

Exit Conditions:

- User removes the song from the playlist

1.4.8 Sorting Playlist (UC-12)

Participating Actor: User

Entry Conditions:

- User has the music player open
- User has a playlist created which has songs within it
- User would like to sort the songs in the playlist

Event Flow:

1. User selects a playlist by clicking it.
2. User then clicks and drags a song to a different position within the same playlist and then releases the mouse button when he has dragged the song to the desired position.
3. Once the song has been dragged to its new location, the user can save the changes permanently by clicking on “Save Playlist”
4. Events 2 and 3 can occur until the user is satisfied with the order of the playlist

Exit Conditions:

- The playlist is reordered and saved

1.4.9 Help (UC-20)

Participating Actor: User

Entry Conditions:

- User has the music player open
- User needs help with some of the music player's features

Event Flow:

1. User selects the “Help” entry in the menu by clicking on it.
2. The “Help” window pops up.
3. The user then peruses the Help document.
4. The user searches for a particular help file by entering the file's name in the “Search” box and then clicks on “Search” to obtain the search results.

Exceptional Case:

- If the help file searched for does not exist, no search results will be returned.

Exit Conditions:

- The user gains access to the “Help” section and obtains the help he needs to use the music player to his satisfaction.

1.4.10 Now Playing (UC-25)

Participating Actor: User

Entry Conditions:

- User has the music player open
- A song from a playlist is currently playing

Event Flow:

1. User wishes to obtain intuitive information regarding the song which is currently playing so he clicks on the “Now Playing” playlist.
2. The music player then brings into view the currently playing song.

Exit Conditions:

- Owner is now informed of the currently playing song as well as all of its metadata

1.4.11 Track List (UC-30)

Participating Actor: User

Entry Conditions:

- User has the music player open

Event Flow:

1. User selects a playlist by clicking on it.
2. The music polayer retrieves the songs ofthe playlist from the database.
3. All the songs from this playlist come into the main view.
4. The user peruses the playlist by scrolling through it with the standard scrollbar on the right side of the window.

Exit Conditions:

- User views the entire list of songs in the given playlist

Exceptional Case:

- If the playlist is empty, the list of songs which come into view will be an empty list.

1.4.12 Active Library (UC-33)

Participating Actor: User

Entry Conditions:

- User has the music player open
- A song from a playlist is currently playing

Event Flow:

1. User wishes to obtain intuitive information regarding the playlist which contains the currently playing song so he clicks on the “Now Playing” playlist.
2. The music player then brings into view the currently playing playlist

Exit Conditions:

- Owner is now informed of the currently playing playlist as well as all of its metadata

1.4.13 Exiting the Player (UC-51)

Participating Actor: User

Entry Conditions:

- User has the music player open

Event Flow:

1. User wishes to close the player so he clicks the “X” at the top right-hand of the window.
2. The “safe close” feature will ensure that all unsaved data gets saved on exit and that all buffers get flushed into the database before terminating the process.

Exit Conditions:

- The player closes

1.4.14 Edit Metadata (UC-56)

Participating Actor: User

Entry Conditions:

- User has the music player open
- User knows which song's metadata he wishes to listen to

Event Flow:

1. The user selects either the entire library or a specific playlist and highlights a track within that list.
2. He then clicks on the “Edit Metadata” button and the “Edit Metadata” window opens.
3. Once in that window, the user edits any of the following items: song title, artist, album, year, track number, genre, rating amd/or custom tags.
4. The user then clicks on “Save”.

Exit Conditions:

- The new metadata is recorded onto the database.

1.4.15 Portability (UC-62)

Participating Actor: Installer

Entry Conditions:

- Installer knows which platform he is installing onto (Windows 32 or Linux)
- Installer has a copy of the installation executable file

Event Flow:

1. The installer double-clicks on the executable (.exe for Windows, .deb for Linux).
2. In the case of Windows 32, the installer will run through all the installation by selecting the desired options and clicking “Next” when appropriate and “Finish” at the end. In the case of Linux, the installer will click on “Install Package” and the player will install itself onto the platform.

Exit Conditions:

- The player will be installed on the system.

Exceptional Case:

- If the installation fails, the installer may remove all (partially) installed files and will attempt to reinstall the player.

2 Architecture

2.1 System Architecture and Subsystems

The system architecture is broken down to the following sub-systems:

2.1.1 Database

The database is the structure that holds a collection of all music files available to the user. In this case, the database is located on the hard disk. Music files can be logically stored in any folder and under any filename. Physically, these files may be stored anywhere on the disk. The physical structure of the disk is split into blocks, and blocks are grouped into sectors. These blocks of data are linked to their logical locations as viewed by the computer system user.

2.1.2 Library

The library is the collection of all music files that have been logically associated with the music player. Essentially, it is a list of all music files that have been added to the music player. It contains each file's information, i.e. file name, location on database, and metadata (artist, title etc). Once a music file is added to the library, it can be accessed directly from the music player's interface. In this way, the user can both select a file from the library to play, or edit the file's information.

2.1.3 Remote Library

This is the same as the library, but it represents the library from another computer system. The music player provides a feature to connect with other users over a network and share playlists. When a user connects with a second user over a network, the second user's library is copied to the first user's remote library. The remote library will then show the exact contents of the second user's library.

2.1.4 Network

The network subsystem is the component of the music player which uses the host computer's network interfaces to connect with other systems on the network, with the purpose of sharing playlists between users. The network sub-system provides the means to access the network user's library and import files from the remote library.

2.1.5 GUI (Graphical User Interface)

The GUI is the interface through which the user interacts with the music player. The GUI creates the music player's appearance on screen as seen by the user. This interface provides the means for the user to perform any action or use any feature of the music player. It includes playback buttons (play, stop, next, previous etc), volume control, library access, file editing and so on.

2.1.6 Playlist

A playlist is a list of music files created by the user. The purpose of this list is for the user to select a group of files for the music player to play. When a user selects to play a playlist, the music player will play all files in this list. All music files added to a playlist are taken from the music player's library. Playlists can also be edited to add or remove music files, and deleted.

2.1.7 Playback

Playback is the sub-system that is responsible for actually playing music files. When a music file is selected to play, this sub-system will find the location of the file, read the data, and then produce it in audio format. The playback sub-system uses GStreamer.

2.1.8 GStreamer

GStreamer is the source code that implements the following:

- reading audio files,
- converting audio formats,
- sending converted data to the audio playback hardware.

Figure 4 illustrates the system architecture and sub-systems. It also depicts the sub-systems interactions and relationships.

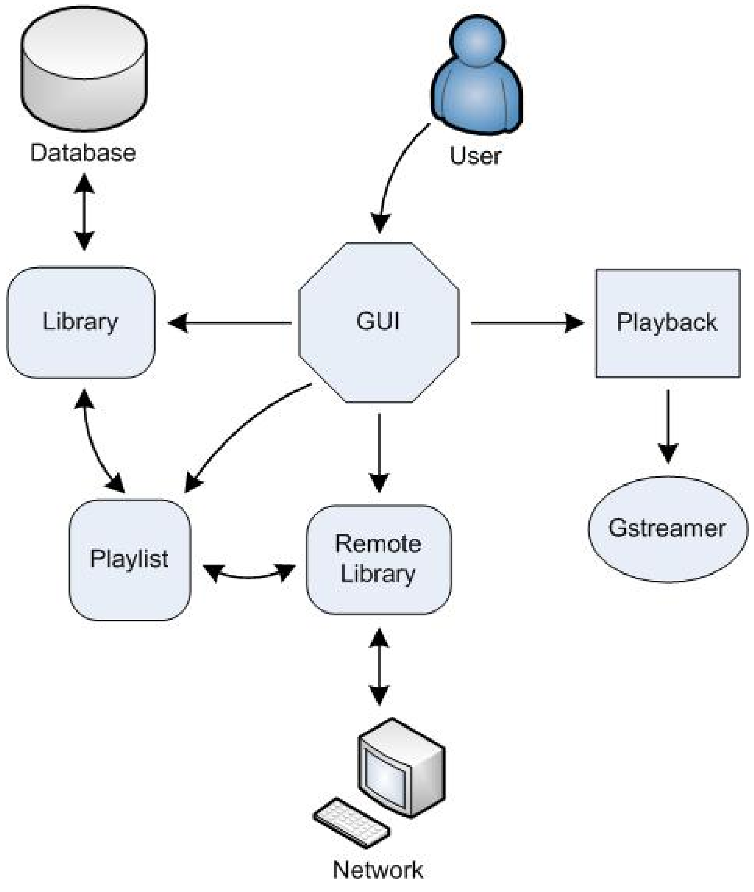

Figure 4: System Architecture

2.2 Subsystem Interactions

The following is an explanatory list of the sub-systems interactions and relationships:

2.2.1 User – GUI

The music player offers several features to the user. To access any of these features, the user interacts with the player's GUI, that is, its interface. The GUI presents the player itself and its features in a certain friendly appearance, including buttons and icons to represent features and actions.

2.2.2 GUI – Library / Remote Library

For the user to access the library or the remote library, the GUI acts as the communications person in between. The user sees the library as the GUI represents it graphically on screen. The GUI also presents buttons and other means necessary for the user to perform actions on the library, such as add, remove or edit music files.

2.2.3 Library – Database, Remote Library – Network

The library has a direct interaction with the database. To add music files to the library, the player accesses the database and retrieves the required music file information. The library holds all the music file information, and once a particular file is selected to play, the library will pass on the location of the file for the player to access it from the database. Similarly, the remote library has the same interaction with the network. The difference in this case is that the network is not a database on the user's computer system. The remote library is the library of another system connected to this one over a network.

2.2.4 GUI – Playlist

The GUI also interacts with the playlist. Again, the user requires an interface to access the playlist, and that is the purpose of the GUI. Through this interface, the user can create or delete playlists, and add or remove music files to a playlist.

2.2.5 Playlist – Library / Remote Library

The playlist is a list of music files created by the user. The purpose of such a list is to instruct the player to play a list of songs rather than only one selected song at a time. The files added to a playlist are taken from the library (or remote library). When a playlist is selected to play, the player accesses the library, which in turn will point to the database and the original storage location of the files.

2.2.6 GUI – Playback

Playback is the sub-system that actually implements the playing of a music file. For the user to interact with the playback, the GUI provides yet another interface to represent available features and actions. Such actions include the basic buttons: play, stop, pause, next etc.

2.2.7 Playback – GStreamer

This interaction makes the playing of music files possible. The playback sub-system controls the GStreamer. To play a song, the GStreamer is instructed to fetch the file data, decode it, convert it to a suitable format and then send it to output.

2.3 Rationale for Chosen Architecture

The system architecture was designed in a way that closely resembles the model-view-controller paradigm. The MVC approach decreases the coupling between each subsystem, as well as improving cohesion (to what extent logically related concepts are grouped in the model).

Strictly following MVC design, however, leads to a couple of disadvantages. For example, the GUI needs to be updated with the name and position of the currently-playing track. If the GUI subsystem were to check periodically for track changes or manage its own updating, it would duplicate the behavior present in the playback subsystem and be tightly coupled to it. On the other hand, if the playback subsystem were to notify the GUI of changes, this would put user interface logic into the controller (lowering cohesion). To avoid this, a variant of the publish/subscribe architecture is used for these kinds of notifications. This is achieved in C# by using delegates and events. In this example, the playback management subsystem provides *events* which any interested participants (in particular, the seek bar and status display portion of the user interface) can subscribe to by adding *event handlers* in the form of delegate callbacks. These events include notifications that the currently-playing track has changed, or periodic notifications to update the position for the currently-playing track.

The different model backends (local and remote) implement a common interfaces that allow for relatively consistent handling from the controller. For example, instead of having logic in the controller which determines how to play a file (i.e. a switch statement for all of the possible file formats, network addresses and protocols over which a track might be played), the track itself has a method which allows the playback management subsystem to ignore these details. The remote library would be able to create concrete instances of a generic interface which transparently handle the details of network streaming. This decouples the playback mechanism from the actual track implementation, and opens up possibilities for future enhancement without too many changes to the architecture (for example, different network protocols, online radio streams, or maybe even voice chat).

The chosen system architecture could be made much more generic, providing support for (e.g.) different widget and windowing toolkits for the frontend, different storage formats for the backend, multiple views for libraries (things like iTunes' Album CoverFlow), etc. The design is however decoupled enough that such changes would not require extensive global changes to the architecture, instead requiring only one component or subsystem to be updated. We deliberately chose to avoid over-architecting these aspects of the music player, since these requirements were not explicitly requested by the customer, and in essence would have been overkill. Although it may require minor modifications during later stages of the design process, the system architecture outlined above meets all of the criteria for an acceptable design: it is simple enough to be feasible and meets all of the customer's major requirements while still being readily changeable.

With respect to the music player's various external dependencies, the C# language along with the .NET platform provides immediate cross-platform support as well as offering several benefits over Java and the JVM; notably, language-level support for database queries (LINQ), event handling, unchecked exceptions, and excellent tooling with Visual Studio or MonoDevelop. GStreamer and GTK, being open source, provide a solid and reliable (as well as legal) backend for the media playback and windowing support, respectively.

3 Static Model

3.1 List of Main Classes

3.1.1 `static class Application`

- Part of the GUI subsystem
- Initializes playback and network subsystems
- Loads and launches GUI
- Stores application-wide globals
- Use Cases:
  - Audio Format (UC-1) – Initializes playback subsystem on local computer with proper parameters to enable all local codecs
  - Initial Setup (UC-46) – Ensures the existence and integrity of the database upon first launch and initial setup
  - Initialization of Services (UC-47) – Initializes the GStreamer pipeline provider, and network Client upon each application startup
  - Exiting the Player (UC-51) – Provides clean exit procedure which is to invoked by the local system upon request for graceful program exit
  - Reliability (UC-60) – attempts to invoke proper cleanup procedure in the event of process termination

3.1.2 `public class MainWindow : Gtk.Window`

- Part of the GUI subsystem
- Represents the main window of the application
- Initializes the Library, Playlist and Track Information viewports
- Initializes the PlaybackStatusDisplay class to provide user with active feedback
- Provides menu bars and hotkeys for common actions
- Use Cases:
  - Documentation (UC-40) – Provides a menu option to launch the product documentation (HTML Format)
  - Backup Library (UC-49) – Provides menu option to export and backup the current library
  - Open Metadata Editor (UC-50) – Provides menu option to launch the metadata editor
  - Intuitive Use (UC-57) – Lays out the components in a fashion similar to the currently prevalent music players on the market, and provides access to product documentation
  - Import Song (UC-14) – Provides menu option to import song or folder containing songs into the library
  - Legality (UC-63) – Provides menu option to view information about the software package, which references class libraries used during the development process and provides links to obtain source code of copy-left libraries
  - Help (UC-20) – The user can launch the product documentation, and then scroll and search through it via functionality provided by the user's browser
  - Volume Display, Equalizer Display (UC-23, UC-24) – The user can view the current settings for the equalizer and other audio filter elements in the pipeline

3.1.3 `public class LibraryView : Gtk.TreeView`

- Part of the GUI subsystem
- Loads list of playlists and libraries from database
- Retrieves information regarding remote playlists from the Remote class
- Provides hierarchical display of music libraries and playlists to MainWindow
- Use Cases:
  - Auto-Sync (UC-19) – allows the user to toggle via a context menu whether the playlist is automatically synchronized with its remote equivalent if the playlist is a remote playlist
  - Location Information (UC-34) – Places appropriate icon next to playlists and libraries located remotely
  - Aggregate Playlist Information (UC-31) – Provides the total time length of each playlist by hovering over it
  - Remote Files Available (UC-37) – Provides a list of remote playlists available
  - Open Metadata Editor (UC-50) – Provides context menu to launch the Metadata Editor window with the currently selected playlist focused
  - Quick Add (UC-55) – Provides for drop location functionality on each playlist (including NowPlaying) so that tracks can be easily added
  - Performance (UC-59) – Loads information via quick SQLite Database backend allowing for minimal lag when program is initiated
  - Playlist Picker (UC-52) – Allows the user to select different playlists at any point in time and have the selection changes reflected within the PlaylistView

3.1.4 `public class PlaylistView : Gtk.TreeView`

- Part of the GUI subsystem
- Loads list of tracks for each playlist selected via the LibraryView from relevant TrackStore
- Provides searchable and sortable list of tracks to user
- Allows for graphical grouping of tracks by album and artist
- Use Cases:
  - Location Information (UC-34) – Places appropriate icon next to tracks located remotely
  - Song Statistics (UC-35) – The user can add a column in detail view to note the number of times each song has been played
  - Track List (UC-30) – provides the user with a list of all the tracks within the selected playlist, with appropriate filters applied
  - Rating (UC-7) – Provides a 1-5 Star Rating widget for assigning a score to each track
  - Aggregate Playlist Information (UC-31) – Displays the sum of the lengths of all tracks in the current view to the user on the top right corner of the panel
  - Remote Files Available (UC-37) – Displays local as well as remote files matching the current filter
  - Metadata Sorting (UC-29) – Allows for songs within the current view to be sorted by metadata for which columns exist
  - Remote Metadata Tags (UC-38) – Displays meta tags for remote files in the same manner as that for local files

    - Open Metadata Editor (UC-50) – Provides context menu to launch the Metadata Editor window with the currently selected song focused
    - Double Click Play (UC-54) – Enables the user to begin playback of a song by merely double clicking it; the current tracks in the PlaylistView are cloned into the NowPlaying list and playback is initialized
    - Quick Add (UC-55) – Provides context menu to add current song to NowPlaying list, and provides drag and drop capability on list items
    - Playlist Viewer (UC-53) – Lists out all the songs within the selected playlist with appropriate filters applied

3.1.5 `public class PlaybackStatusDisplay : Gtk.VBox`

- Part of the GUI subsystem
- Provides access to information conveying the current playback status, such as song length, and current position
- Provides information regarding the active song, such as title, album, and artist name
- Provides a trackbar which indicates current position, and can be modified to alter current position within a song (seeking).
- Use Cases:
    - Playback Control (UC-2) – Allows the user to seek within the song as necessary
    - Active Track (UC-27) – provides information on the currently active track
    - Artwork Viewing (UC-28) – Displays album artwork of active track during playback

3.1.6 `public class PreferencesWindow : Gtk.Window`

- Part of the GUI subsystem
- Use Cases:
    - Automatic Metadata Management (UC-9) – provides GUI option to enable this feature
    - Network Song Access (UC-16) – provides the user with a ComboBox widget for storing a list of network locations to be explicitly searched for media
    - Share Playlists (UC-18) – provides radio button option to enable sharing, and ComboBox for list of shared locations

3.1.7 `public class MetadataEditor : Gtk.Window`

- Part of the GUI subsystem
- Use Cases:
    - Edit Metadata (UC-56) – Provides user a list of all the current metadata which is associated with the selected track, and provides text boxes for permanent modification of the information
    - Artwork Editing (UC-8) – Enables the user to replace the album artwork associated with the currently selected track
    - Custom Metadata (UC-6) – Provides custom tags field to the user for storage arbitrary metadata to be associated with the active track
    - Metadata Manager (UC-5) – Provides the GUI for the user to create, update, or delete metadata associated with each song
    - Rating (UC-7) – Allows the user to revise the rating associated with each song

    - Automatic Metadata Management (UC-9) – can sift through songs in the PlaylistView, automatically looking up metadata information through an internet connection
    - Metadata Synchronization (UC-43) – Allows for the metadata displayed in the PlaylistView to be automatically refreshed when the user saves a change to the file via this editor

3.1.8 `public class Configuration`

- Part of the GUI subsystem
- Stores the user's custom configuration information to be loaded each time the program starts
- Use Cases:
    - Automatic Metadata Management (UC-9)– Stores the user's setting for this option
    - Network Song Access (UC-16) – Stores a list of locations for network searches
    - Share Playlists (UC-18) – Stores the users preferences regarding media sharing
    - Favorite Friends (UC-39) – Stores a list of remote locations to be always visible within the LibraryView panel
    - Database Loading (UC-45) – Stores the location of the local library database
    - Initialization of Services (UC-47) – Stores the user's preferences regarding which features are available and should be loaded upon program startup
    - Consistent Gain (UC-22) – Stores the users volume and equalizer settings prior to program exit so that user's playback configuration remains consistent even through multiple startups
    - Default Library (UC-15) – Stores the location of the users primary music library for automatic loading on startup

3.1.9 `public interface ITrack`

- Part of the Music Storage subsystem
- Provides a generic interface for storing song meta-information
- Can be stored in the database to quickly retrieve information regarding the song regardless of actual storage location
- Use Cases:
    - Playback Control (UC-2) – Provides interface for obtaining GStreamer audio source

3.1.10 `public interface ITrackStore : ICollection<Track>`

- Part of the Music Storage subsystem
- Provides a generic interface for storage of an arbitrary collection of songs
- Use Cases:
    - Search (UC-13) – Enables filtering functionality by enumerating through items in the current store via a given search delegate function
    - Sorting Playlist (UC-12) – Provides method to sort items in the current store via a given comparator delegate function

3.1.11 `public interface ILibrary : ITrackStore`

- Part of the Music Storage subsystem
- Represents a music library stored in an arbitrary location

- Provides a generic interface for storing a collection of songs and playlists without particular ordering
- Contains members storing information regarding the host system and user to which the library is localized
- Use Cases:
    - Backup Library (UC-49) – Can be saved to Database or XML File format for backup purposes
    - Import Song (UC-14) – Allows for tracks to be added to the current Library given a file or folder location

3.1.12 `public interface IPlaylist : ITrackStore`

- Part of the Music Storage subsystem
- Provides a generic interface for storing a collection of songs with particular ordering and repetitions allowed
- Contains members storing meta-information pertaining to the entire playlist
- Use Cases:
    - Playlist Add (UC-10) – Allows for user to add tracks to the current playlist
    - Playlist Remove (UC-11) – Provides method which can be invoked to remove a track from the current list
    - Sorting Playlist (UC-12) – Enables ordering changes via ordering provided by underlying TrackStore class sorting method

3.1.13 `public class LocalLibrary : ITrackStore`

- Part of the Music Storage subsystem
- Represents a collection of playlists and songs which are stored on the local computer
- Use Cases:
    - Default Library (UC-15) – The default library shall be a LocalLibrary object

3.1.14 `public class LocalTrack : ITrack`

- Part of the Music Storage subsystem
- Represents a song which is stored on the local machine
- Implements ITrack to enable playback and storage of meta-info into a database
- Provides a GStreamer Audio Source object compatible with the playback manager
- Use Cases:
    - Playback Control (UC-2) – Provides GStreamer Audio Source object referencing the local file for playback

3.1.15 `public class NowPlaying : IPlaylist`

- Part of the Playback subsystem
- Maintains a collection of songs which are playing during the current session
- Implements IPlaylist to provide common methods to save and load the playlist from the database
- Exposes properties for configuration options such as shuffle and repeat

- Hooks into OnTrackFinished event to select, load, and play the next track in the play queue based on shuffle and repeat settings
- Use Cases:
    - Now Playing (UC-25) – Provides the playlist which is to be mapped to the Now Playing playlist in the library view panel.
    - Current Order (UC-32) – Provides method to automatically clones songs in the playlist view, allowing for ordering to be temporarily modified.
    - Active Track (UC-27) – Provides information regarding the currently active track
    - Intuitive Use (UC-57) – Enables easy instant playlist creation by maintaining a separate list of songs to be played within the current session
    - Auto Playback Statistics (UC-44) – Automatically updates the relevant play count for each track as it is activated

3.1.16 `public class PlaybackManager`

- Part of the Playback subsystem
- Interfaces with GStreamer backend to provide audio output
- Receives Audio Source object from currently playing track
- Constructs audio pipeline retrieving data from the relevant audio source
- Manages audio filters on the audio pipeline
- Publically exposes playback events to other objects:
    - OnStateChanged – fires when playbackstate has changed
    - OnPositionChanged – fires when the current position in the track has changed
    - OnPlaybackTick – fires at regular intervals during audio playback
    - OnTrackChanged – fires when the loaded track changes
    - OnTrackStarted – fires when a track has begun playback
    - OnTrackFinished – fires when a track has reached the end of playback
- Use Cases:
    - Audio Format (UC-1) – Interfaces with GStreamer which automatically handles audio codecs.
    - Playback Control (UC-2) – Provides wrappers around GStreamer methods to provide stop, pause, and seek functionality.
    - Volume Control & Equalizer (UC-3, UC-4) – Places appropriate GStreamer filters on the pipeline
    - Network Song Access (UC-16) – Provides remote Audio Source to GStreamer for remote playback
    - Playback State (UC-26) – Publically exposes properties and triggers events related to playback state
    - Background (UC-48) – GStreamer automatically detects proper audio output devices on the host system
    - Quality Assurance (UC-21) – Configures GStreamer to enable uninterrupted high quality media playback

3.1.17 `public class Client`

- Part of the Network subsystem
- Provides the connectivity to the peer to peer network via the XMPP presence protocol
- Responds to peer requests regarding shared media libraries as per user configuration
- Connects to peers to inquire about shared media
- Provides a mechanism to spawn new RemoteLibrary objects from external media
- Use Cases:
  - Remote Song Import (UC-17) – Transfers remote song locally and adds to library, or simply adds remote network location to library as per the user's specification
  - Security (UC-61) – Prevents access to songs and other data not explicitly shared by the user

3.1.18 `public class RemoteLibrary : ILibrary`

- Part of the Network subsystem
- Represents libraries not residing on the local computer
- Provides listing of available (shared by the peer) playlist and tracks
- Provides GStreamer Audio Source objects for remote streaming of songs
- Provides method to transfer content of remote songs to local computer
- Use Cases:
  - Crawler (UC-41) – Automatically updates the contents of remote playlists when queried
  - Auto-Sync (UC-42) – Automatically manages list of remote media contents and keeps local information up to date
  - Metadata Synchronization (UC-43) – allows locally stored metadata for remote files to be automatically refreshed upon remote modification of this information

3.1.1 `public class Installer`

- Part of the deployment subsystem
- Represents a set of rules to be used during deployment in the windows platform
- Extracts necessary dependencies and adds them to a location visible to the user's system path
- Use Cases:
  - Initial Setup (UC-46) – Allows the user to install and initialize the setup the music player on the local system
  - Legality (UC-63) – Provides an End User License Agreement which must be agreed to prior to software installation
  - Portability (UC-62) – Compiles to an executable which will unpack and install the necessary to avoid unnecessarily complicated deployment procedure on the windows platform.
  - Package (UC-58) – As part of the installation procedure, the help documentation is to be extracted and placed on the users machine for later retrieval

3.2 Class Diagrams

To illustrate the interconnectivity required between staccato and the GStreamer class library, the following diagram demonstrates some of the association relationships driving the Staccato PlaybackManager. A DecodeBin is a GStreamer object which automatically deciphers the incoming audio format of the `src` object specified. A GStreamer Element object is one with some combination of `sinks` and `pads`, which can be linked together in a `pipeline`. This provides sufficient connections to decode and playback audio. Any desired audio filters, such as volume and equalizer controls, can also be placed on this pipeline.

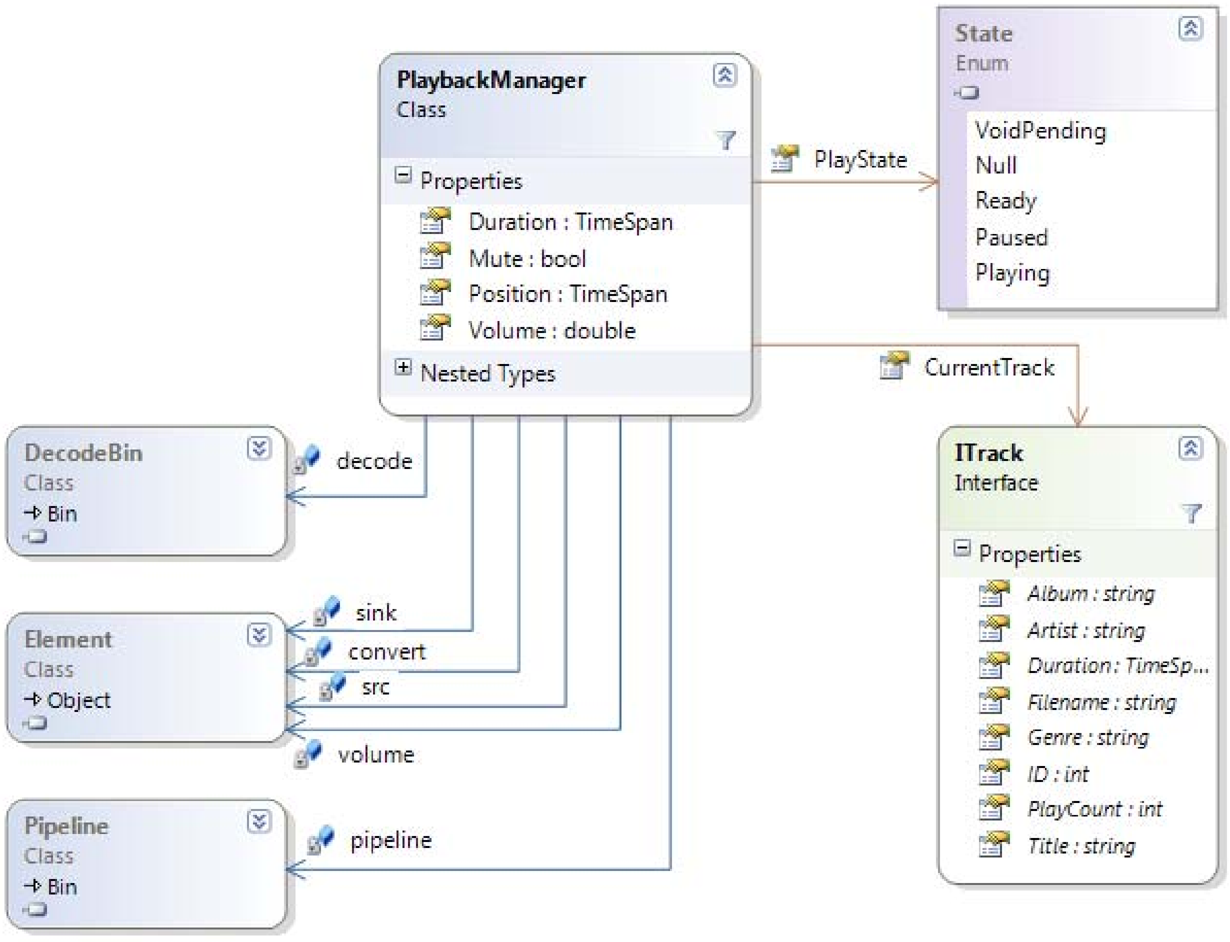

Figure 5: Static model for playback subsystem

The following diagram illustrates the association relationships linking together the concepts of Tracks and Playlists as stored in the database and represented by Entity objects in memory. The database is kept in sync with the state of the Entity objects in memory via functionality provided by the DBLinq ORM, which extends C#'s LINQ functionality.

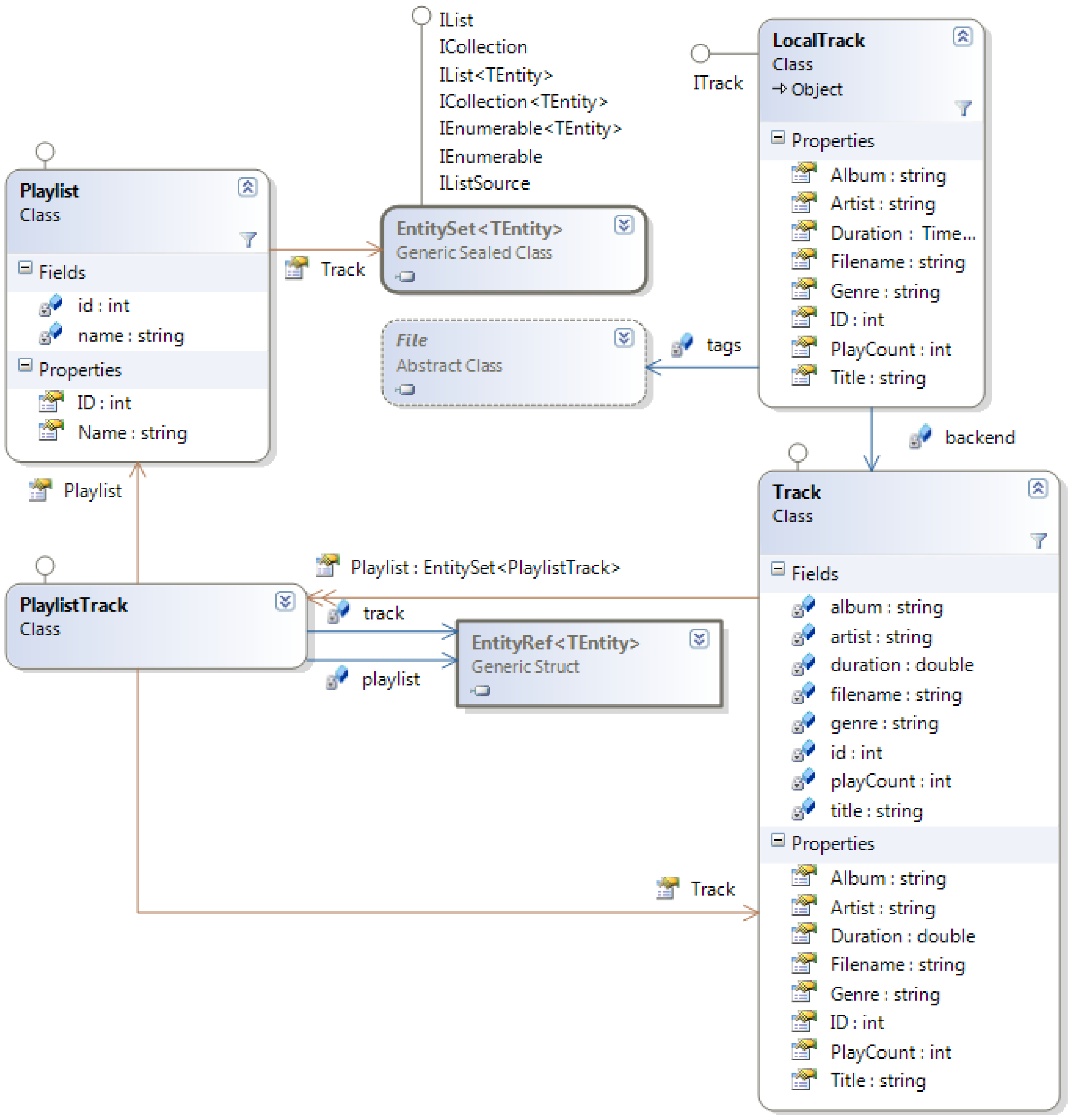

Figure 6: Static model for database library entities

The following diagram provides a generic overview of the classes involved in providing a Graphical interface in the software. A majority of the GUI classes inherit from GTK classes and are only modified slightly to provide the necessary functionality.

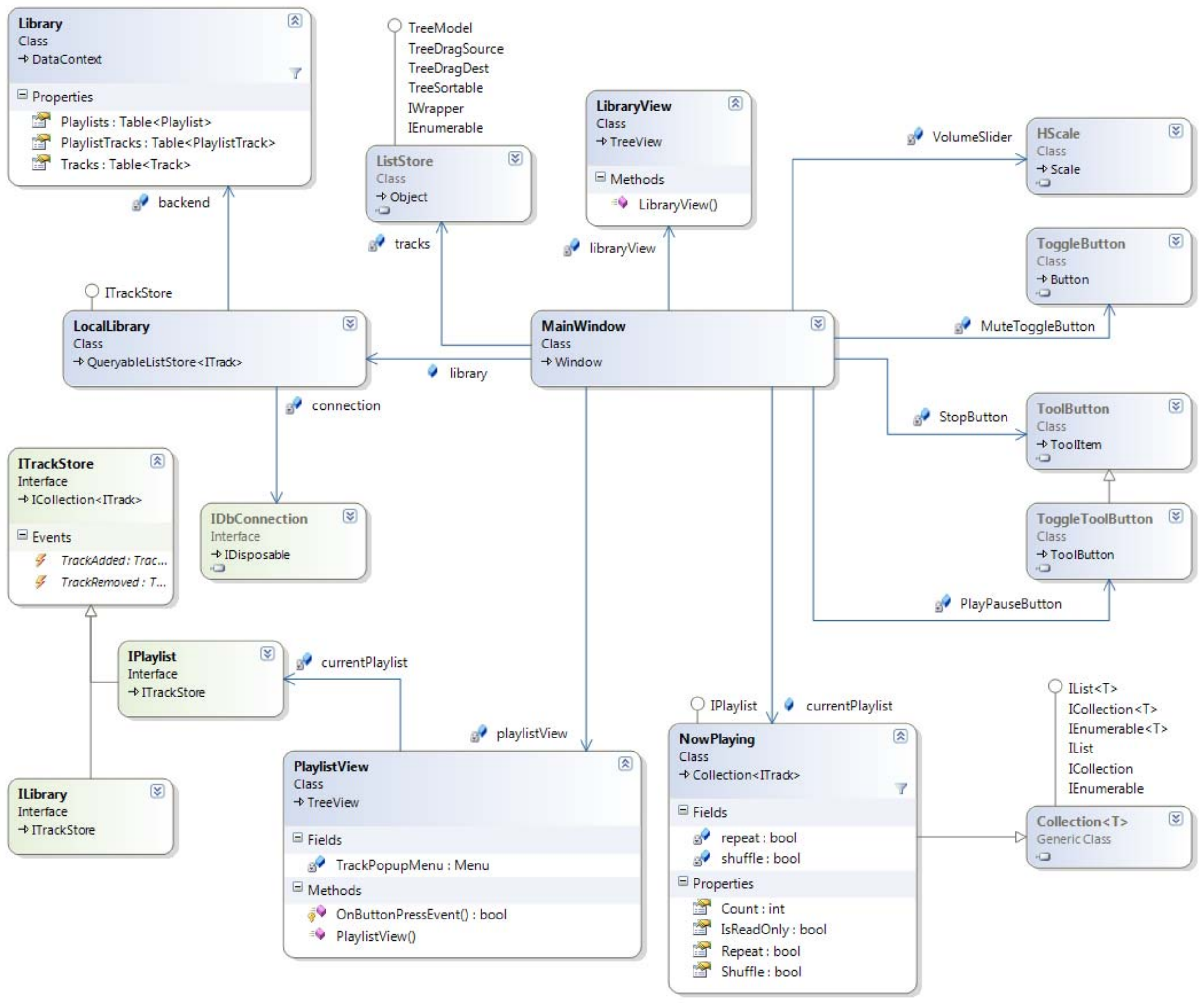

Figure 7: Static model for GUI subsystem

4 Test Plan

4.1 List of Test Scenarios

Test #	Affiliated use Case	Test preconditions	Required actions and expected results	Actual Results	Pass/Fail
S1	UC-1: The **user** wants to play files encoded in one of the major audio formats.	Have an OS installed on a machine that is powered on. Have access to the required codec files and an audio file of presently unrecognized format.	Install the appropriate audio codecs. Afterwards, the player will recognize the audio files.		
S2	UC-2: The **user** is listening to a song, and he wants to stop, pause, or seek.	Have a song playing on the player.	Press on the appropriate buttons on the GUI (Triangle = play, square = stop, 2 vertical lines = pause, 2 triangles = seek). The song's playback status is updated accordingly.		
S3	UC-3: The **user** is listening to a song and wishes to modify the volume of the player's audio output.	Have a song playing on the player.	Slides the volume slide bar or press the mute button. The volume level by the player is updates accordingly.		
S4	UC-4: The **user** wishes to modify the gain of specific audio frequencies outputted by the player.	Have a song playing on the player.	Click on the equalizer button in the main window and slides the equalizer scales representing the band on which the gain is to be changed.		
S5	UC-5: The **user** wishes to create, update or delete metadata associated with an audio file (i.e. song title, album, cover,	Have the song visible in the main window.	Select the song in the main window and click the "edit metadata". A window will open with all the esitable fields. Press save and the metadata will be upgraded.		

	etc…).				
S6	UC-6: The **user** wants to create his own personal fields of metadata and name them.	Have the song to which the new metadata is to be applied visible and selected in the main window.	Click on the appropriate menu item and a dialogue opens.		
S7	UC-7: The **user** wants to set a rating he sees fit for particular songs.	Have the song to which the new rating is to be applied visible and selected in the main window.	Choose "Edit Rating" in the menu function. Edit the rating and save.		
S8	UC-8: The **user** wants to change the artwork associated with an audio file.	Have the song to which the artwork is to be modified visible and selected in the main window.	Select the appropriate menu button and update the image file.		
S9	UC-9: The **user** wants the player to automatically manage the metadata associated with the songs.	Have the player running and songs in the main library who's audio files contain metadata.	Enter that choice in the configuration option of the menu. The player will extract the metadata it can from the audio file and update it on the player.		
S10	UC-10: The **user** wishes to add songs to a playlist.	Have a playlist created and ready to accept new songs, and some songs in the main library which are not already on that playlist.	Click the song from the main view and drags it to the playlist.		
S11	UC-11: The **user** wishes to delete a song from a playlist.	Have a playlist containing at least one song.	Select the playlist such that it is visible in the main window. Selects the undesired song and click on it such that it is highlighted, then presses "delete" then "save playlist".		
S12	UC-12: The **user** wants to change the	Have a playlist visible I the main view, containing at least 2	Click and drag the desired songs up and down the main view. To		

	order in which the songs are played within a playlist.	songs.	make the changes permanent, click on the "save playlist" button.
S13	UC-13: The **user** wants to search for a particular song within a playlist.	Have the desired song present in any playlist, not necessarily in the main view.	Enter the song name in the search field and presses "search" button. The playlist containing the song will appear in the main view with the song highlighted.
S14	UC-14: The **user** can import a file/directory on the local hard drive by opening the "import window" from the menu.	Have some audio files present on the file system which are not already in the player's music library. (Note, the appropriate audio codecs need to have been installed prior to import).	Importing an audio file will cause that file to be added to the library. Importing a directory will cause all audio files contained recursively within it to be added to the library.
S15	UC-15: The **user** wants to have a default import library for the songs imported.	Have the player running and some songs to be imported.	Enter the configuration panel via the menu and chooses default library.
S16	UC-16: The **user** wishes to select which network location will be browsed for songs.	Have some network drives available on the file system and containing audio files.	This is done via the configuration panel, accessed from the menu.
S17	UC-17: The **user** can select specific files and click on the "import" button to import those files to his local song library.	Have the import dialogue open and pointing at a directory containing audio files.	Select specific files and click on the "import" button
S18	UC-18: The **user** can select whether or not he wishes to	The user is viewing the configuration view (accessed from the menu) and is part of a	Select the appropriate option in the configuration and click save.

	share particular songs/playlists.	computer network.	
S19	UC-19: The **user** wants one of his playlist to synchronize automatically with a remote repository.	The user is viewing the configuration view (accessed from the menu).	Select that option in playlist configurations.
S20	UC-20: The **user** needs help with some of the software's features.	The player must be running.	Select the help entry in the menu and a help window pops up. Scroll the document and use search function.
S21	UC-21: The **user** wants to listen to undistorted music and not have to deal with signal analysis.	Have the music player as well as a well recognized one (i.e. Itunes, QuickTimes or WMP) both running and pointing to the same audio file.	Play the audio file with Stacatto then with the other player and compare the audio quality. There should be no (or minimal) difference between both sounds.
S22	UC-22: The **user** wants to hear songs at a stable level without having to continuously readjust volume and equalizer	Have a playlist containing many songs from different sources.	Listen to all the songs in the playlist. There should not be any major gain distortions audible. (Note, even if there are minor gain variations, if they are not noticeable by human ear, they can be neglected)
S23	UC-23: The **user** wants to be able to see what the volume is set to within the main view.	Have the player running.	Look at the top right corner of the player. There should be a scroll bar indicating the volume. Move it left and right. It should be representative of the audio output level.
S24	UC-24: The **user** wants to be able to see the current frequency gain settings before	Have the equalizer dialogue open.	Look at the gain level and change them. A solid correlation between the visual position of the bands gain and the audio gain should be

	updating them.		observable.
S25	UC-25: The **user** wants to get intuitive information the track playing.	Have a song playing with associated metadata.	Should be able to know the position of the song, as well as name and associated metadata within the main window.
S26	UC-26: The **user** wishes to get information on the current playback state.	Have a song playing.	Look at the "time left" at the bottom of the player. If the song is playing, the time remaining is shrinking.
S27	UC-27: The **user** wishes to see the Title, Album and Artist associated with the current track playing.	Have a song playing with the appropriate metadata.	Look at the top of the player. The metadata should be visible.
S28	UC-28: The **user** wishes to see the Artwork associated with the file currently playing.	Have a song playing to which there is an associated image file.	Look at the top right corner of the player. The image file should be displayed.
S29	UC-29: The **user** wants to be able to sort his playlist based on a metadata item.	Have a playlist containing at least 2 songs with a common metadata field.	Click sort by "meta tag" and the list of songs within the visible playlist are sorted by the meta Tag selected.
S30	UC-30: The **user** wishes to see all the tracks within a playlist.	Have a non empty playlist which is not currently displayed in the main view.	Click on the desired playlist in the playlist view. The list of tracks will appear in the main view, allowing for easy navigation using a standard sidebar.
S31	UC-31: The **user** can learn the total number of songs within a	Have a playlist created in the player.	Double click on the playlist in the playlist dialogue such that the songs it contains become visible in the main view.

	playlist as well as its total duration.		The number of songs as well as total duration will be displayed at the bottom of the main view.		
S32	UC-32: The **user** wishes to temporarily change the order of songs within a playlist.	Have a playlist containing at least 2 songs.	To view the current order, he can click on "now playing" and will see a clone of the playlist he originally selected to play.		
S33	UC-33: The **user** wishes to see all the songs within the active library by clicking on "now playing".	Be listening to songs within a library while it is not displayed in the main window.	Click on “now Playing”. This content of the active library will be displayed within the main window.		
S34	UC-34: The **user** wishes to see whether the music files present on the playlist are located on his local machine on at a remote location.	Have songs located on a local machine as well as songs located on a remote computer.	Select the playlist containing the songs and the relevant information will be displayed within the main window.		
S35	UC-35: The **user** can get playback information on any songs present within his library.	Have a music library containing songs.	Select the library in the library dialogue. Select the view entire library tab and looking at the appropriate column.		
S36	UC-36: The **user** wishes to see a list of available remote locations from which we may extract songs.	Have network access to remote file systems containing audio files supported by the player.	Select view remote locations from the menu function. A list of remote locations should appear.		

S37	UC-37: The **user** can click on a remote location which is available, and a list of available songs will appear within the main window.	Have network access to remote file systems containing audio files supported by the player.	Click on a remote location which is available, and a list of available songs will appear within the main window.		
S38	UC-38: The **user** will also see all the available metadata in the main view.	Be viewing a song playlist from a remote location in the main view.	After the remote playlist has been selected, the list of songs as well as their metadata will be displayed in the main view.		
S39	UC-39: The **user** wishes to specify a list of known remote network location for future quick access.	Have network access to remote file systems.	Open the remote locations panel and select the appropriate option. Restart the computer and open the player. The remote location should still be visible.		
S40	UC-40: The **user** wishes to consult documentation relevant to the player.	Have the player running.	Select the "help" function in the menu.		
S41	UC-41: The **user** wants the player to keep an updated list of available remote resources on the network without having to continuously refresh and track down location manually.	Have network access to remote file systems containing audio files supported by the player as well as physical access to that host's computer, and have a remote playlist displayed within the main view.	Modify the remote playlist from the host machine. Shortly after the modifications take place on the remote machine, they should appear on the local view of the remote playlist.		
S42	UC-42: The **user** wants the	Have network access to remote file systems,	In the options dialogue, select which remote		

	player to synchronize automatically with certain pre-determined remote locations.	as well as physical access to that host’s computer.	locations to automatically sync. With. The run test S41.	
S43	UC-43: The **user** wishes to change the metadata associated with a song.	Have a song visible in the main window	Open the edit metadata dialogue and change some values. The player automatically updates the audio file to reflect that change. Vice versa, whenever a song is refreshed, its metadata is read from the file and displayed on the screen.	
S44	UC-44: The **user** doesn't have to do anything in particular for playback statistics to get updated. The playback statistics are automatically updated and displayed.	Be viewing a playlist in the main window.	Add a song to the playlist and observe the playback statistics. They will be automatically updated and displayed to reflect the changes.	
S45	UC-45: When the music player is started by the **user**, the default music library's information is automatically fetched from the DB.	Have already created playlist and added songs to the player, and not have the music player running.	Launch the player. The default music library's information is automatically fetched from the DB.	
S46	UC-46: When the **installer** installs the music player on a computer, a database is	Have a computer on which the music player is not installed.	Install the music player on that computer. A database is automatically created to store library information. It can be tested by adding audio	

	created to store library information.		files to the main library.
S47	UC-47: When the **user** loads the music player, the audio filters, network utilities are set up. The player also caches data relevant to the music library to speed up future operations.	Have a computer on which the music player is installed but not running, and playlists and songs have previously been created/added to the player's DB.	Launch the player. The audio filters, network utilities are set up. The player also caches data relevant to the music library to speed up future operations. Observe the speed of operation, and the fact that audio files can be played.
S48	UC-48: The **user** does not have to worry about any kind of background operations involving the host environment.	Have the player running.	Any of the tests performed will not require knowledge of background operations; only an understanding of the high level GIU functions.
S49	UC-49: The **user** can export the contents of his library by selecting this menu item. The backups created will be importable from any other players.	Have a player with a song library containing at least one song on one computer. On a second computer, have a player installed with audio files on the disk, but nothing in the main library.	The user selects the export library menu item. A dialogue will open to name and save the back up file. After the backup is made, import it from a second machine, not linked to the first one.
S50	UC-50: The **user** wishes to open the metadata editor window.	The player must be running.	Select this menu item. When clicked, a separate window will open (see metadata Window below for more details)
S51	UC-51: The **user** can close the player using this button. A	The player must be running, and not minimized.	Make modifications to the library and click the close window button. The player should close,

	safe close will ensure that all unsaved data gets saved and that buffers get flushed to the database before terminating the process.		and flush to the DB any unsaved buffers before terminating the process. The verify this, reopen the player and observe that the changes made in the previous session are still present.		
S52	UC-52: The **user** can toggle between viewing different playlists and the main song library using this area.	Have at least one playlist created with content different from the main library.	Selecting different playlists/main library items in the library window will toggle the main view between the contents of those playlists.		
S53	UC-53: The **user** can see all the songs present within a playlist, as well as their respective metadata in this area.	Have at least one playlist created and containing songs with metadata.	When selecting that playlist in the playlists view, all the songs it contains will be visible in the main view, as well as their metadata.		
S54	UC-54: The **user** can play a song simply by double clicking on it in the playlist viewer.	Have at least one song visible in the main view which is not currently playing.	Double click on the song and it will start playing.		
S55	UC-55: The **user** can add a song to a playlist by selecting it in the playlist viewer and dropping it in the desired playlist within the playlist picker.	Have a playlist created. Have a song in the main view which is not present in that playlist.	Click on the song and drag it to the desired playlist. Then, by selecting that playlist in the playlist view, its content will appear in the min window, updated to reflect the newly added song.		

S56	UC-56: The **user** can open the metadata editor window from the menu after highlighting a track. Once open, the **user** will be able to edit the song title, artist, Album, Year, Track number, Genre, Rating, Custom Tags. After editing, pressing the "save" button will record the new data within the database.	Have the player running and containing at least one song.	Highlight a song and open the metadata editor window in the menu. Edit values, press “save”. The changes should be reflected in the main view, in the metadata display field of those songs.	
S57	UC-57: The **user** should be able to figure out how to use the music player intuitively, based on previous experience with other players. Otherwise, full documentation is available to describe its functions.	Be a user who has never used the player before, but is familiar with WMP or Itunes.	The new user opens the player and should be able to create playlists, add songs to them, play/pause/seek songs without any external help.	
S58	UC-58: When installing the software, in addition to the music player, the **installer** will install help documentation.	Have a computer on which the player is not installed.	Install the player. The help documentation should also get installed.	

S59	UC-59: The **user** can load a playlist containing 1000 songs within 1 second on a computer meeting the minimum system's requirements of the player.	Have the main library in the main view, and have previously created a playlist containing 1000 songs.	Select the 1000 song playlist from the playlist view, and the playlist's content should be displayed in the main window within one second.		
S60	UC-60: The **user** can terminate the music player process abruptly without any effect on the audio files and data base content on the local/network drives.	Have the player running on a PC or a laptop without a battery.	Cut the power supply of the computer. Plug it back in and load the music player. Audio files and playlists should not be corrupted.		
S61	UC-61 The **hacker** will not be able to access the user's data if the user did not select the share option for that data.	Have a computer connected to a network and the player running with the option "share music" unselected. Also have access to a second machine on the same network.	Through the player installed on the second machine, try to access files from the first computer. Access should be blocked.		
S62	UC-62: The **Installer** will be able to install the player on a Windows 32 platform using the executable, or on a Linux platform using the .deb package.	Have a computer running Windows and a second one running Linux.	Install the player on both machines using the .exe for Windows and the .deb for Linux. The player should install on both machines.		

S63	UC-63: The **lawyer** will not be able to sue the software developers on the basis of Canadian computer software regulations.	Have the player released to the public.	If we don’t get sued, the test is a success.		

4.2 Thoroughness of test plan

We have tested 62/63 of use cases, and since there is one use case associated with every requirement, we have simultaneously tested 62/53 requirements. The only use case note tested is UC-63, which would require legal advice above our budget of 0$ for this project.

Te most important tests were those related to the playback of audio files, because if a music player is not able to play audio files, it is not a music player. Every other requirements are extra features, which distinguishes Staccato from other music player.

4.3 Failure Analysis

FILL ME!!

FILL ME!!

FILL ME!!

FILL ME!!

FILL ME!!

FILL ME!!

FILL ME!!

For failed tests, is there an assessment of why the test fails? What are the symptoms? Possible leads

5 Requirement Traceability Matrix

The table below provides the requirement traceability matrix for the use cases presented in

Use Case Model as well as the classes in the Static Model back to the requirements in the updated requirements specification document submitted along with this document.

Requirement	Use Cases	Classes	Achieved
3.1 – Functional Requirements			
3.1.1 – Inputs			
3.1.1.1 – Audio Output Management	UC-1, UC-2	Application PlaybackStatusDisplay ITrack LocalTrack PlaybackManager	Yes
3.1.1.2 – Audio Filter Manager	UC-3, UC-4	PlaybackManager	Yes
3.1.1.3 – Metadata Manager	UC-5, UC-6, UC-7, UC-8, UC-9	PlaylistView PreferencesWindow MetadataEditor Configuration	Yes
3.1.1.4 – Playlist Manager	UC-10, UC-11, UC-12, UC-13	ITrackStore IPlaylist	Yes
3.1.1.5 – Music Library Manager	UC-14, UC-15	MainWindow Configuration ILibrary LocalLibrary	Yes
3.1.1.6 – Network Manager	UC-16, UC-17, UC-18, UC-19	LibraryView PreferencesWindow Configuration PlaybackManager Client	Yes
3.1.1.7 – Software Assistance	UC-20	MainWindow	Yes
3.1.2 – Outputs			
3.1.2.1 – Audio Output	UC-21, UC-22	Configuration PlaybackManager	Yes
3.1.2.2 – Audio Filter Manager	UC-23, UC-24	MainWindow	Yes
3.1.2.3 – Music Player State	UC-25, UC-26, UC-27	PlaybackStatusDisplay NowPlaying PlaybackManager	Yes
3.1.2.4 – Metadata Manager	UC-28, UC-29	PlaylistView PlaybackStatusDisplay	Yes
3.1.2.5 – Playlist Manager	UC-30, UC-31, UC-32	LibraryView PlaylistView NowPlaying	Yes
3.1.2.6 – Music Library Manager	UC-33, UC-34, UC-35	LibraryView PlaylistView	Yes
3.1.2.7 – Network Manager	UC-18, UC-36, UC-37, UC-38, UC-39	LibraryView PlaylistView Configuration	Yes
3.1.2.8 – Software Assistance	UC-40	MainWindow	Yes

3.1.3 – Services			
3.1.3.1 – Network Manager	UC-41, UC-42	RemoteLibrary	Yes
3.1.3.2 – Metadata Manager	UC-43, UC-44	MetadataEditor NowPlaying RemoteLibrary	Yes
3.1.4 – Initialization			
3.1.4.1 – Database	UC-45, UC-46, UC-49	Application MainWindow Configuration ILibrary Installer	Yes
3.1.4.2 – Services	UC-47, UC-48	Application Configuration PlaybackManager	Yes
3.1.5 – User Interface			
3.1.5.1 – Main Window	UC-50, UC-51, UC-52, UC-53, UC-54, UC-55	Application MainWindow LibraryView PlaylistView	Yes
3.1.5.2 – Metadata Editor Window	UC-50, UC-56	MetadataEditor	Yes
3.1.5.3 – Import Window	UC-14	MainWindow ILibrary	Yes
3.1.5.4 – Help Window	UC-58	Installer	Yes
3.2 – Non-functional Requirements			
3.2.1 – Usability	UC-57	MainWindow NowPlaying	Yes
3.2.2 – Performance	UC-59	LibraryView	Yes
3.2.3 – Reliability	UC-60	Application	Yes
3.2.4 – Security	UC-61	Client	Yes
3.2.5 – Portability	UC-62	Installer	Yes
3.2.6 – Legality	UC-63	MainWindow Installer	Yes

Table 1: Requirements Traceability Matrix