# Object-Based Playlists

A Senior Project

presented to

the Faculty of the Computer Science & Engineering

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science

by

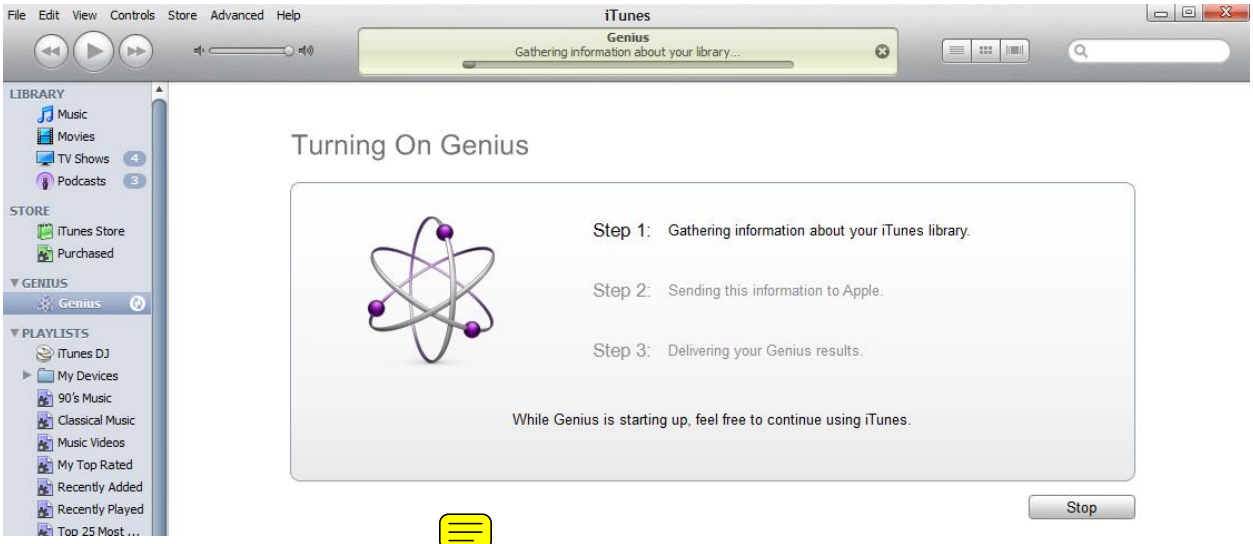
Aaron Baldwin
Jeremy Bradshaw
Kevin Mathew

June, 2011

# Introduction

## Problem

Playlists originated several decades ago from radio stations. The stations needed to limit and keep track of the listing of songs that played daily. Today, a playlist refers to an order of songs within a time period. Even though unique methods of constructing playlists have been developed, such as iTunes' Genius or Smart Playlists, the main issue at hand is that the core structure remains unchanged.




With smart playlists like Genius, the user does not have complete control over what each playlist contains. Smart playlists analyze a user's library or current playlist content. Based on the data accumulated, it will decide and generate playlists on the go for users which is convenient. However, the user may not agree with the selections from the smart playlist.

| | | | | | |
|---|---|---|---|---|---|
| 968 | Who Killed Davey Moore? | 3:09 | Bob Dylan | The Bootleg Series Vol. 1-3 [Rare & Unreleased] 1961-1991 [Disc 1] | Folk |
| 969 | Who The Cap Fit | 4:17 | Bob Marley | Natural Mystic | Reggae |
| 970 | Who The Cap Fit | 4:17 | Bob Marley | Natural Mystic | Reggae |
| 971 | Who The Cap Fit | 4:43 | Bob Marley | Rastaman Vibration | Reggae |
| 972 | Who The Cap Fit | 4:43 | Bob Marley | Rastaman Vibration | Reggae |
| 973 | 01 - Guess Who Is Back (Intro) - Boh | 1:50 | Bohemia | The Punjabi Invasion | Pop |
| 974 | 01 - Guess Who Is Back (Intro) - Boh | 1:50 | Bohemia | The Punjabi Invasion | Pop |
| 975 | The Girl Who Stole the Stars | 3:49 | Chrono Cross Soundtrack | Chrono Cross | Game |
| 976 | The Girl Who Stole the Stars | 3:49 | Chrono Cross Soundtrack | Chrono Cross | Game |
| 977 | Little Queenie | 2:43 | Chuck Berry | The Great Twenty-Eight | Rock |
| 978 | Little Queenie | 2:43 | Chuck Berry | The Great Twenty-Eight | Rock |
| 979 | Anyone Who Had A Heart | 2:51 | Cilla Black | The British Invasion (1963-1967) (Disc 1) | Rock |
| 980 | Anyone Who Had A Heart | 2:51 | Cilla Black | The British Invasion (1963-1967) (Disc 1) | Rock |
| 981 | Who's Got My Back? | 8:26 | Creed | Weathered | Rock |
| 982 | Who's Got My Back? | 8:26 | Creed | Weathered | Rock |
| 983 | Beauty Queen | 3:45 | CzarNok | Coach Carter | Soundtrack |
| 984 | Beauty Queen | 3:45 | CzarNok | Coach Carter | Soundtrack |
| 985 | Beauty Queen | 3:45 | CzarNok | Coach Carter | Soundtrack |
| 986 | Beauty Queen | 3:45 | CzarNok | Coach Carter | Soundtrack |
| 987 | Sea of Simulation | 2:40 | Daft Punk | Amazon Bonus Track | Soundtrack |
| 988 | Father and Son | 3:16 | Daft Punk | iTunes Bonus Track | Soundtrack |
| 989 | Outlands Pt.II | 2:57 | Daft Punk | iTunes Bonus Track | Soundtrack |
| 990 | Sunrise prelude | 2:50 | Daft Punk | Nokia Ovi Store Bonus Track | Soundtrack |
| 991 | Overture | 2:28 | Daft Punk | Tron: Legacy (Cd1) | Soundtrack |
| 992 | The Grid | 1:37 | Daft Punk | Tron: Legacy (Cd1) | Soundtrack |
| 993 | The Son of Flynn | 1:35 | Daft Punk | Tron: Legacy (Cd1) | Soundtrack |
| 994 | Recognizer | 2:38 | Daft Punk | Tron: Legacy (Cd1) | Soundtrack |
| 995 | Armory | 2:03 | Daft Punk | Tron: Legacy (Cd1) | Soundtrack |
| 996 | Arena | 1:33 | Daft Punk | Tron: Legacy (Cd1) | Soundtrack |
| 997 | Rinzler | 2:18 | Daft Punk | Tron: Legacy (Cd1) | Soundtrack |
| 998 | The Game Has Changed | 3:26 | Daft Punk | Tron: Legacy (Cd1) | Soundtrack |

Playlists can contain numerous amounts of songs. Organizing and keeping track of all these songs within a playlist ~~is~~ difficult. Because there are so many songs listed, the playlist can look messy or like a wall of text that seams together in one blur. Some media players do not catch or check for duplicate entries. If one attempts to combine two or multiple playlists together, the problem is still there. Lists of all the songs from all of the playlists involved in the combination are listed. The user cannot keep track of what song came from what playlist on a large scale.

## Solution

The answer to the problem is hierarchical playlists, or also known as object-based playlists. The basic premise of an object-based playlist is that each item within a playlist is a playlist itself, allowing greater complexity and control of the playlist format. Playlists, at their core, have remained the same since their inception: a list of songs to be played in or out of order. Advancements such as "smart playlists" only make the creation and filling of playlists easier by guessing at what the user wants to listen to using sample data. This system still uses the same core playlist mechanic.

Object-based playlists resemble the structure of classic file systems. This new type of playlist allows for a granular control of playlists. More complex structures can be created without the mess. An example with a mess would be combining two playlists. With this system, a playlist can be listed within a playlist.

Playlists are no longer treated as simple lists of songs to play. Playlists are considered to be a 'media object.' Playlists are not the only items to be considered as a media object. Essentially, everything is which include songs, albums, and artists. With this system, playlists are now a list of media objects which opens up more robust methods of creating a playlist.

In Figure 1, the UML diagram of the relationships between playlists, artists, albums, and songs is shown. All of these are considered to be interfaced with 'MediaObject.' A playlist contains entries of MediaObjects which can mean anything. Artists contain lists of albums by that specific artist. Albums contain songs within the album. The song is the deepest child in the tree and is the object that is actually played.

Figure 2 shows a good, simple example of the capabilities of object-based playlists. With an object-based playlist, a user can create a playlist containing all the albums from his or her favorite band (one entry on the playlist per album), set the music player to random, and then one album is randomly chosen to be played, in order, in its entirety, then another album is picked, and so on. The figure below illustrates the structure of the described situation:

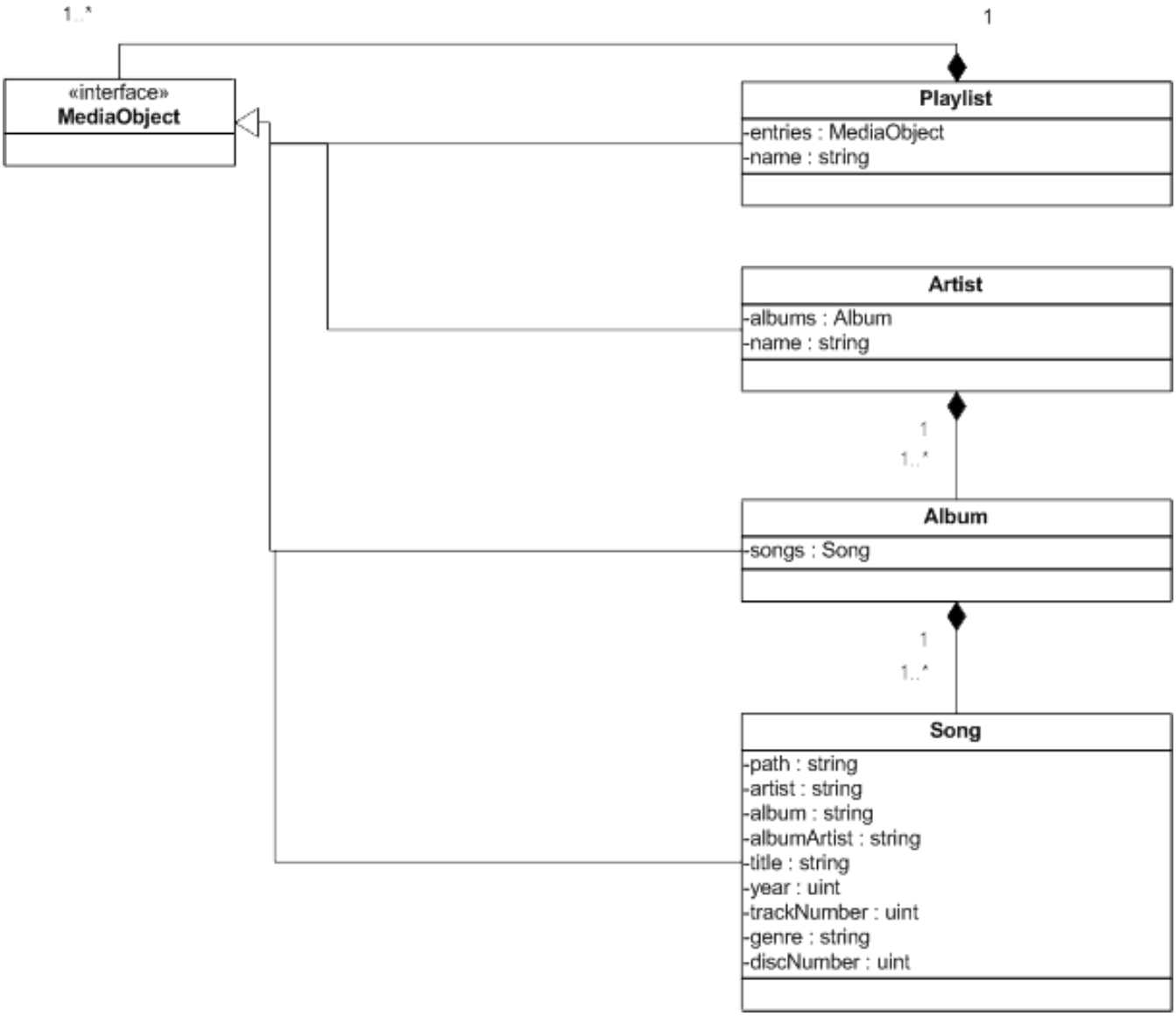



*Figure 1: UML Diagram of hierarchical system*

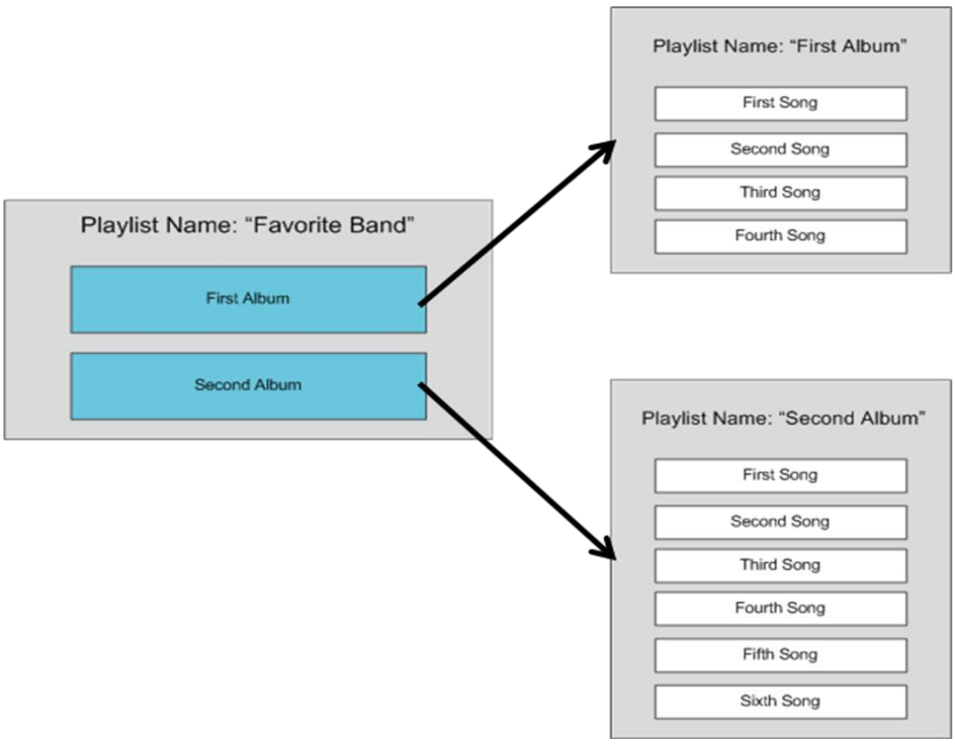



*Figure 2: A playlist containing 2 playlists as entries*

Figure 2 illustrates a very simple example, but figure 3 shows a slightly more complex model using the same object-based playlist architecture. The "Good" playlist in figure 3, if played in order, will play "First Playlist" first, which plays "Another Playlist" before the last song in "First Playlist." Then the song in the "Good" playlist will play, immediately followed by "Second Playlist."

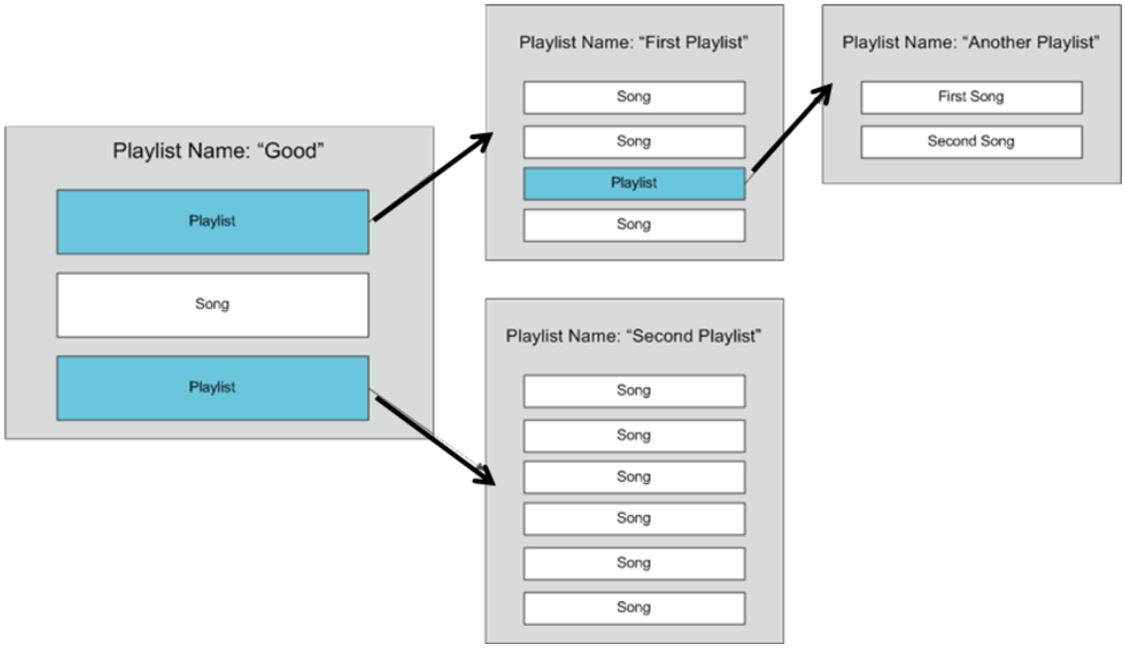



*Figure 3: A more complex playlist*

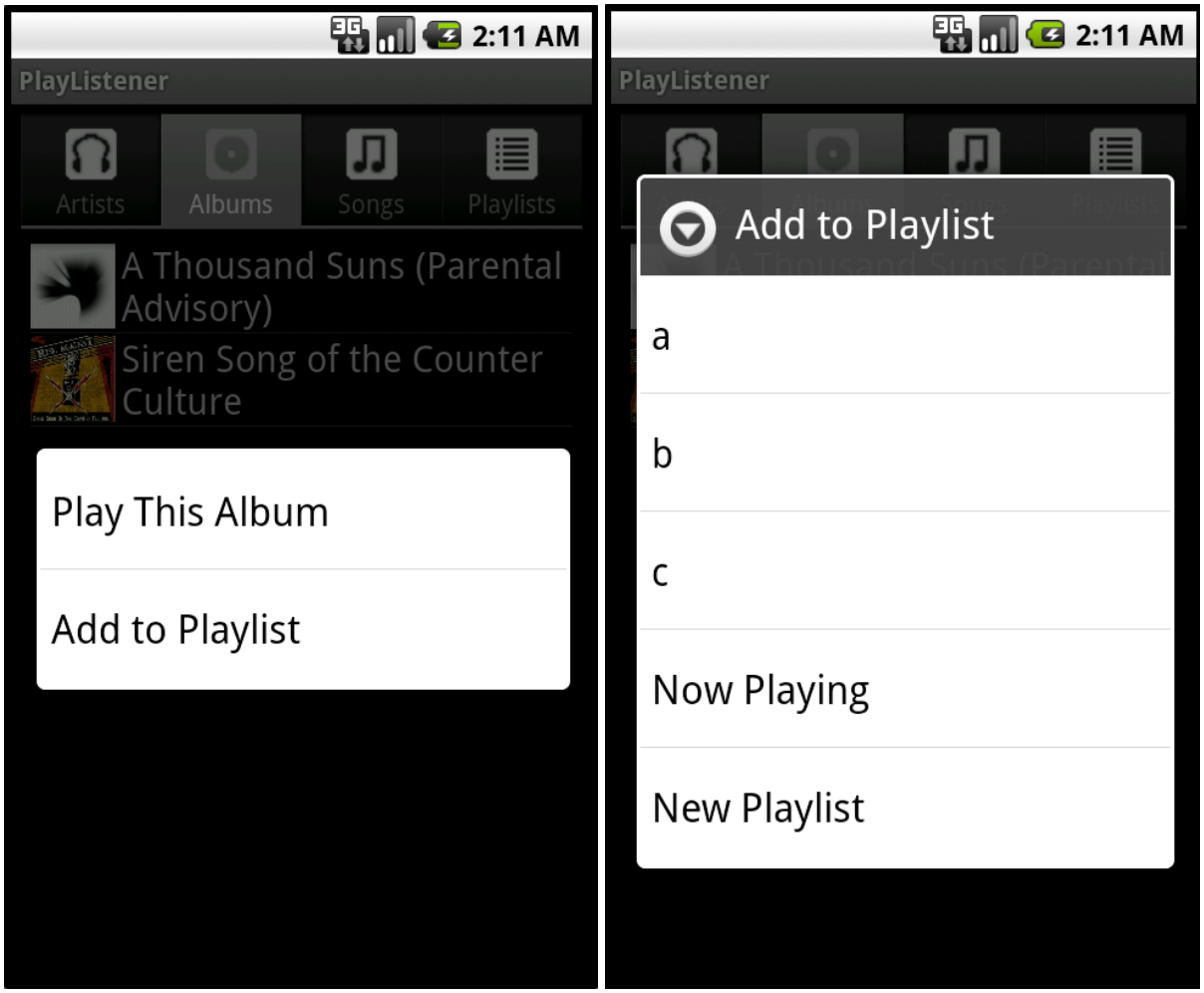
**Outline**

Will add when report is mainly finished

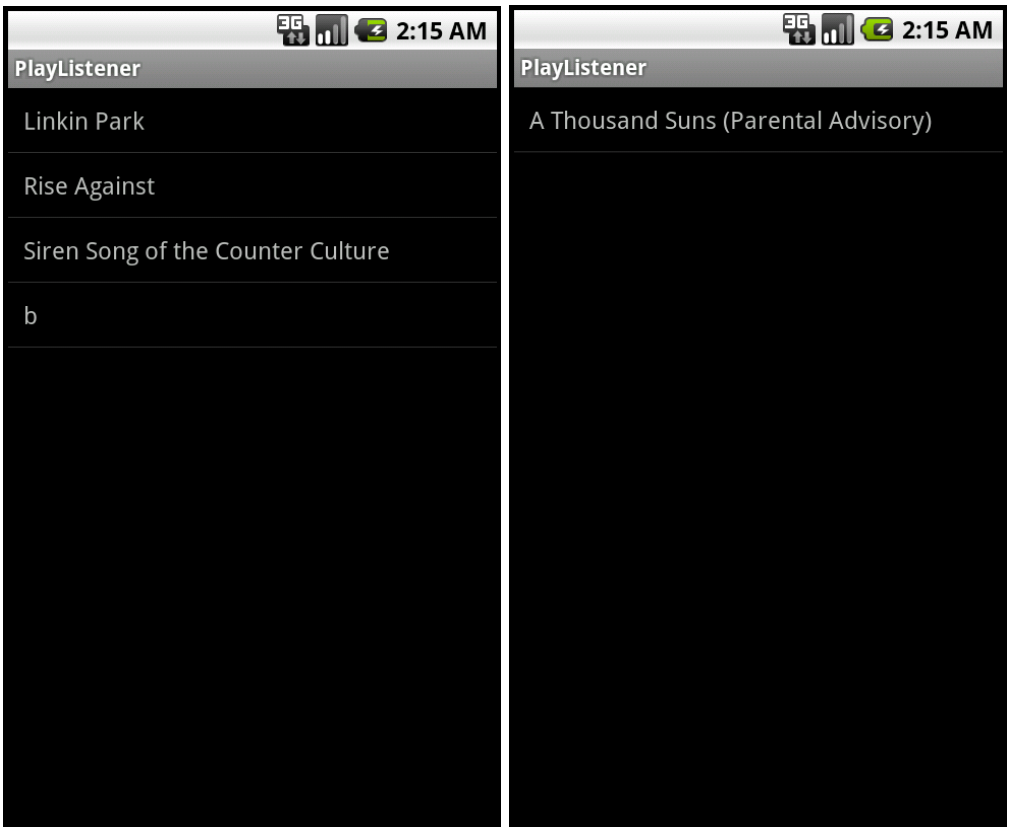
Objective

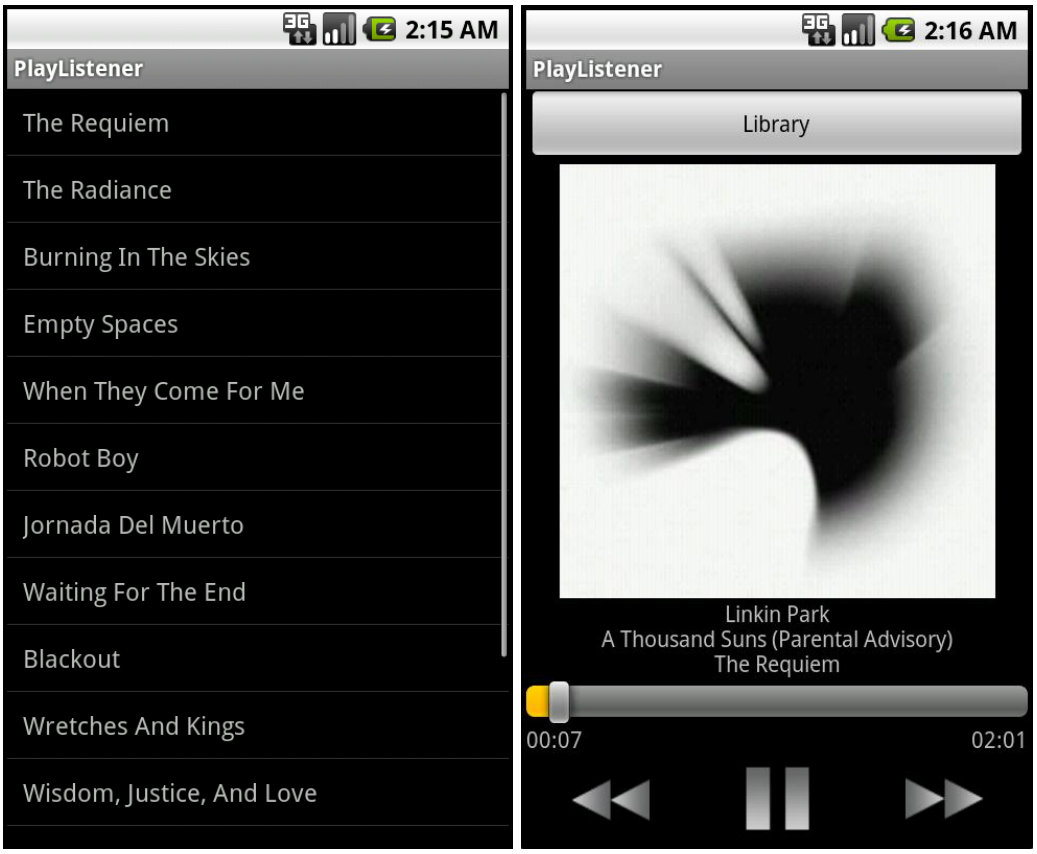
## Scenario of System Use

The main interesting feature of this project is the ability to customize playlists the way the user wants. The user is able to add Media Objects to either existing playlists or create a brand new playlist.



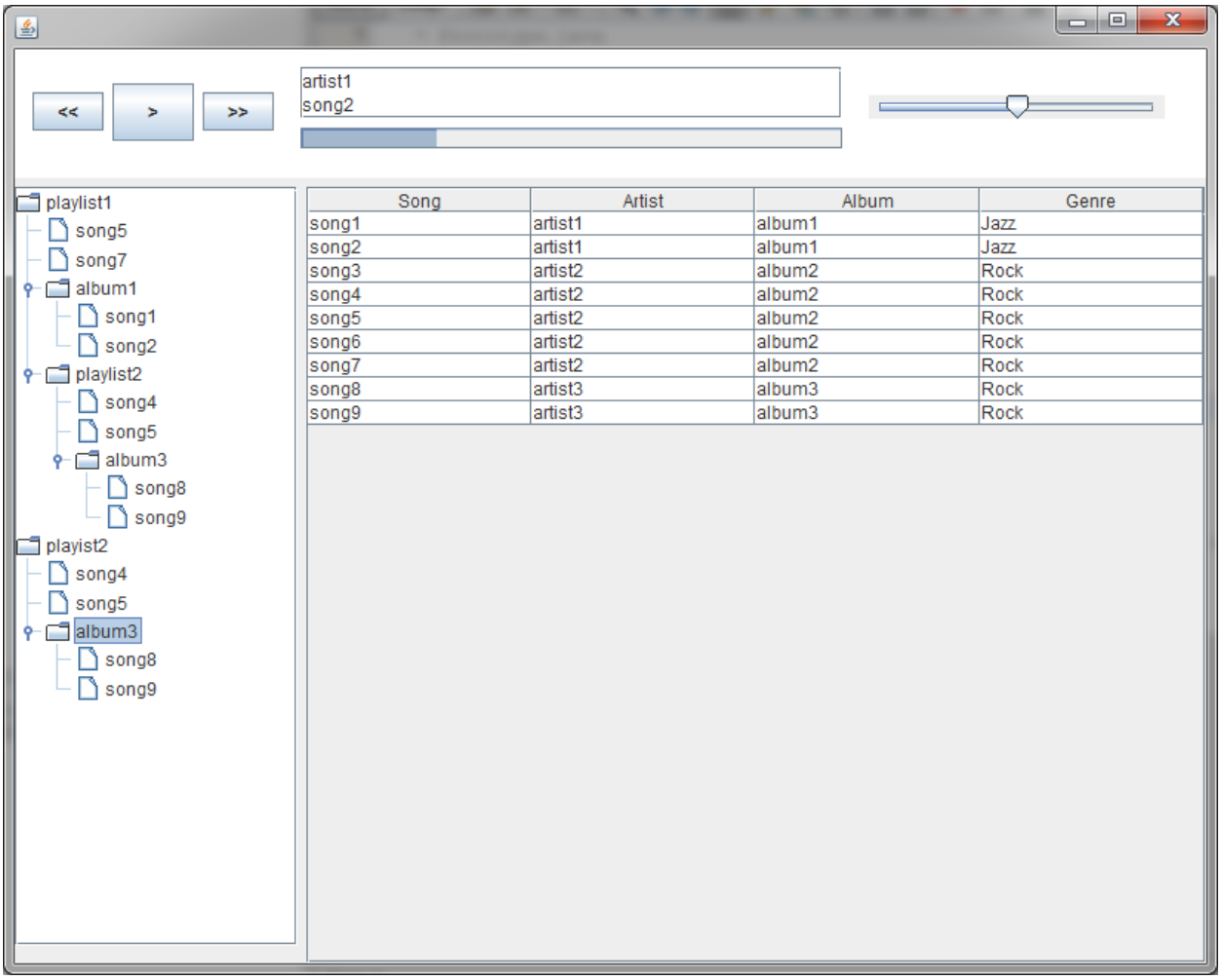

The user has the option under the playlist tab to edit currently existing playlists. They can also traverse through a selected playlist. In the following example, the user selects playlist ‘a’ and then selects ‘Linkin Park.’ The user gets taken to a list of all available Linkin Park albums. The album that displays is ‘A Thousand Suns.’ The user can venture even further down the tree by selecting the album. Now all the songs listed under the album are displayed. When playing the playlist, the media player will go down the tree to where the songs are and play them. Once all the songs are played, it will go up to its parent and move to the next branch.


2:15 AM
PlayListener
Linkin Park
Rise Against
Siren Song of the Counter Culture
b
2:15 AM
PlayListener
A Thousand Suns (Parental Advisory)




## Design & Implementation

The original platform choice was to have the system work on the desktop. A simple mock GUI was created with Java Swing. The basic desktop GUI was able to play music and it was able to display a hierarchy for the playlists. However, the application did not operate in the way a normal user would want. There were extreme delays during skipping songs and pausing/playing them as well. This became a serious issue. The decision to switch to another platform was made in order to allow more focus on the core of the project which is the structure of how the playlists and songs are listed.




The alternative platform selected was Android. The reasoning behind this is because the project originally began in Java. Android is a Java based operating system for mobile devices developed by Google. The SDK is for Android is readily available for download and is not difficult to set up. It also is easy to setup with Eclipse. The application is codenamed 'PlayListener.'

//need more on evolution of android app

## Testing

The Android SDK includes the ability to run an emulator of the different versions of the operating systems. PlayListener was tested on Éclair, Android 2.1, and Froyo, Android 2.2. Gingerbread was not tested because there are not as many mobile devices with that specific version. Its predecessors are far more popular and saturated within the consumer based market.

On the ‘Now Playing’ screen, the things that need testing are the following:

| Function | Evidence |
|---|---|
| Pause/Play Button | Switches icon between pause/play<br>Pauses/plays current song |
| Next Button | Switches to next song within queue or list |
| Previous Button | Switches to previous song within queue or list |
| Seek Bar | Seek bar moves at appropriate speed based on length of song.<br>Moving seek bar can lead to skipping through or rewinding through songs properly. |
| Album Art | If song is playing, album art should display if song is properly tagged or has the art. |
| Library Button | Leads to Library screen with tabs for different display lists of songs. |
| Menu Button | Displays the option to exit and quit the application. |
| Exit Button | Exits application. |

For the ‘Library’ screen, the functionality needing testing are the following:

| Function | Evidence |
|---|---|
| Tabs | When selecting a tab, the appropriate section should display in the content viewer |
| Select Song | Song should immediately play on now playing |
| Select Album | List of songs within album should display in content viewer. |
| Select Artist | List of albums by artist should display in content viewer. |
| Hold Down on Song, Artist, Album | Ability to add object to existing playlist.<br>Create a new playlist and add an object to it.<br>Add the object to the queue for now playing. |
| Create New Playlist | Customizable name.<br>OK and Cancel buttons work.<br>Playlist displays under playlist tab in library. |
| Hold Down on Playlist | Play playlist.<br>Add additional objects to playlist.<br>Edit criteria for the playlist.<br>Set the playlist to repeat.<br>Shuffle the elements of the playlist.<br>Rename the playlist.<br>Delete the playlist. |
| Menu Button | Displays the option to exit and quit the application. |
| Exit Button | Exits application. |

## Technical Section III

## Related Work

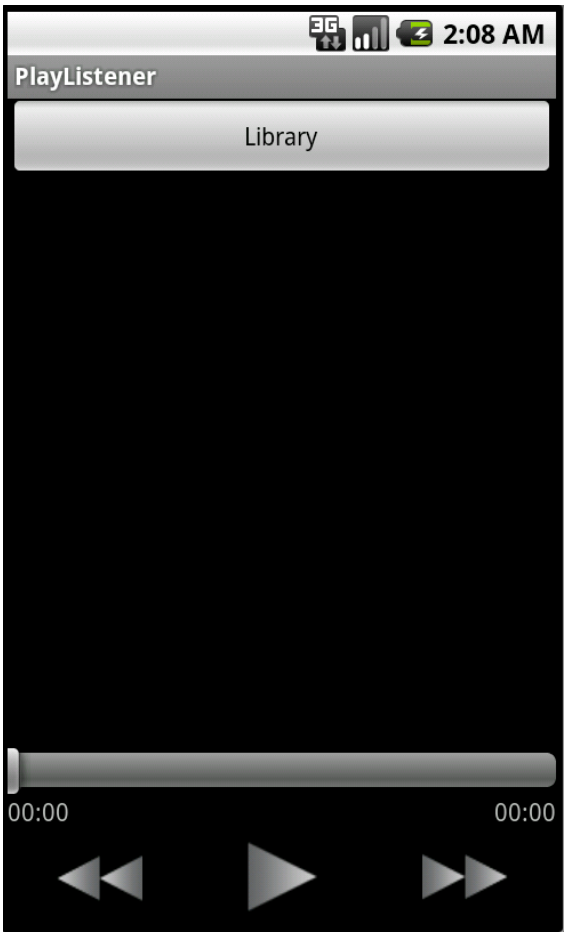
## Conclusion & Future

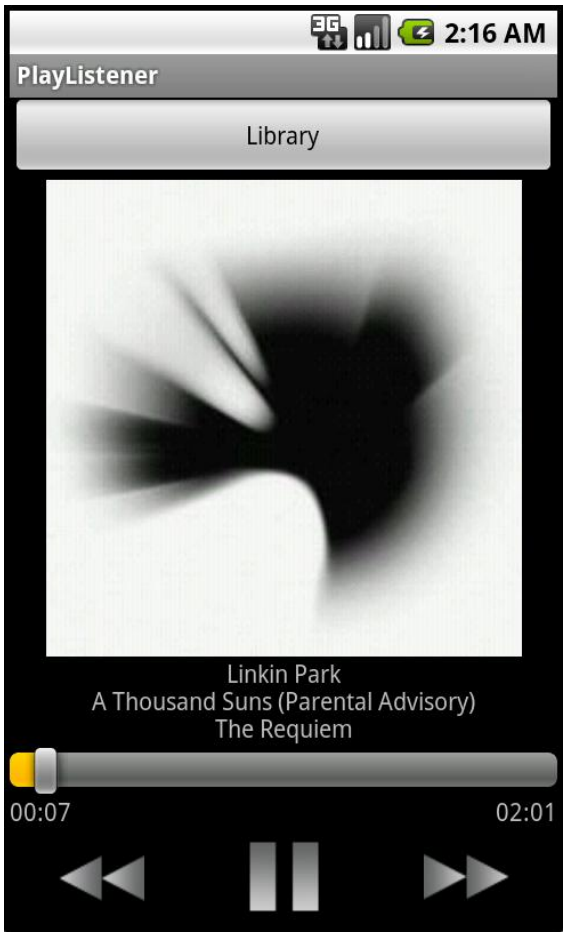
Currently, the android application functions properly. The main purpose behind driving the project works. The user is able to customize the way they want their playlist structure to look like. The user can continuously create nested playlists or list all their favorite albums. However, the 'PlayListener' application is still not available on the market. There are still a few issues regarding crashes on alternate versions of the operating system.

Now, there are plans to move the core model view design of the playlist structure to various platforms. There is talk of creating a desktop application that will allow users to sync their media to their mobile devices. Because this was not the focus of the project, these ideas have been pushed for work in the future.

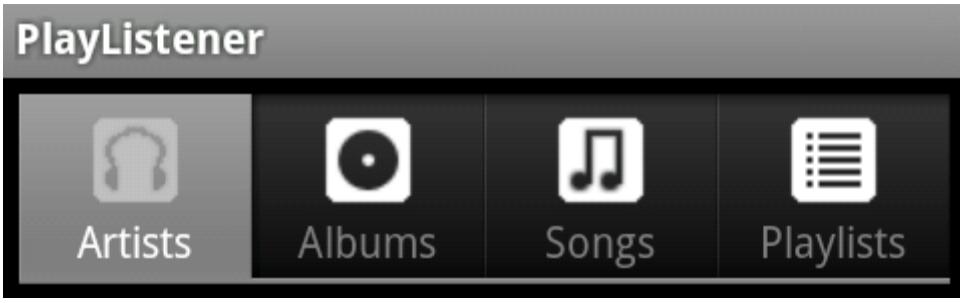
## Appendix A: User Manual

When first launching the app, the user is greeted with the Now Playing screen. On this screen, the user has control over the seek-bar to traverse through the song and buttons to pause/play or skip through the Now Playing queue. The album art of the current song is displayed above the seek-bar.



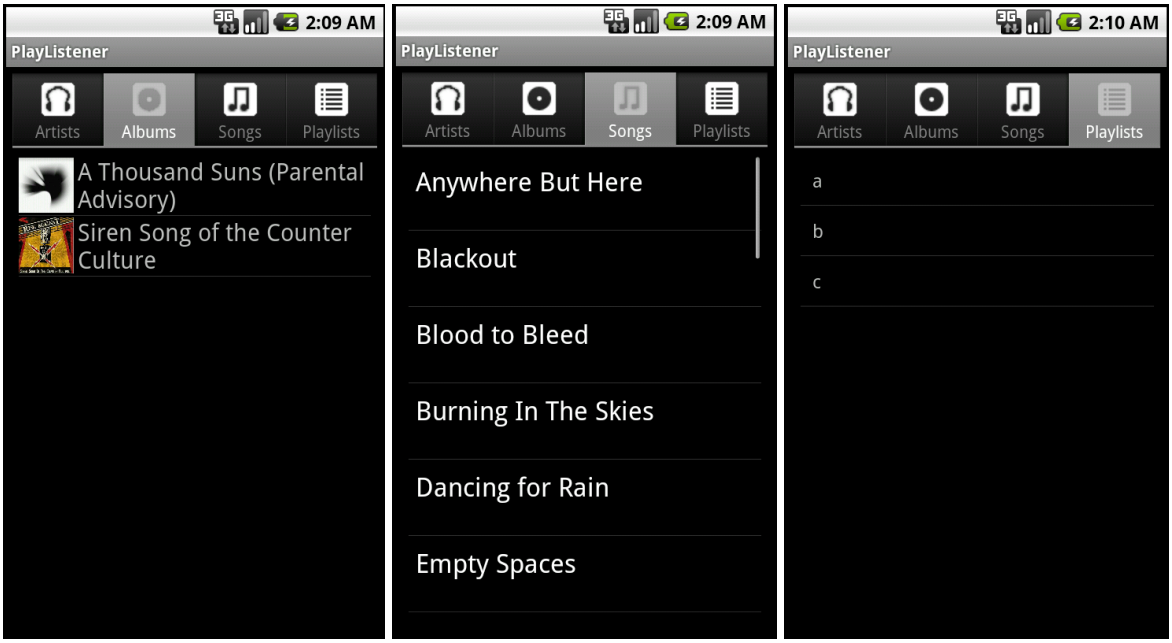



In order to select a song, the user must first touch the Library button located near the top. The Library screen will now show up. The Library section is navigated with the use of the tab system. The tabs sort the songs according to the following criteria: Artists, Albums, Songs, and Playlists.



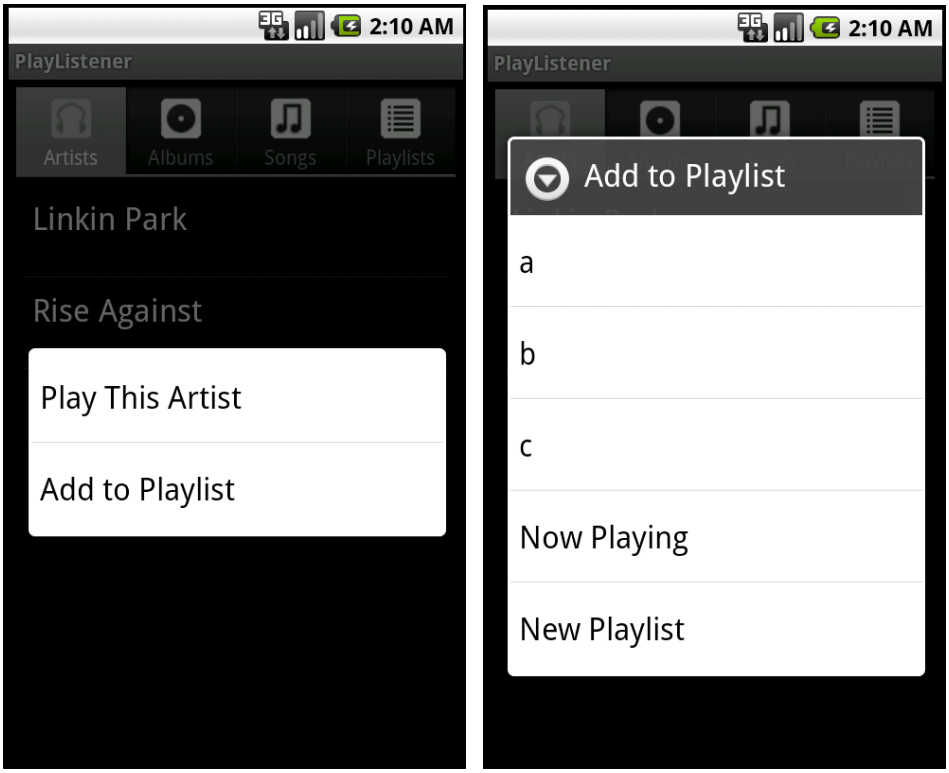

Each of these sections contains a list of the appropriate Music Objects depending on the current tab that is selected. The artists tab lists all artists. When selecting one of the artists, a new screen will appear with all the albums on the device done by the artist. The user can decide on what album they want to play, or they can go even further to by looking for a specific song under one of the albums.

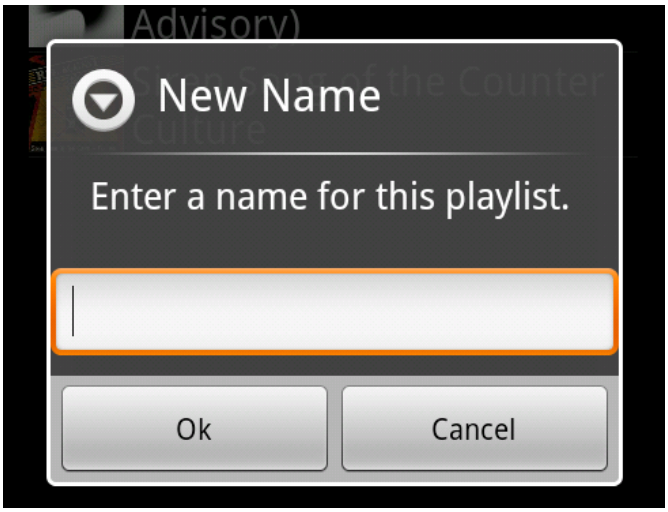
Under the albums tab, a list is displayed of all the available albums on the mobile device. Alongside each of the listed albums to the left of the title is the album art. This allows the user to experience a visual recognition of what each album is which may allow the user to distinguish the different albums from one another.



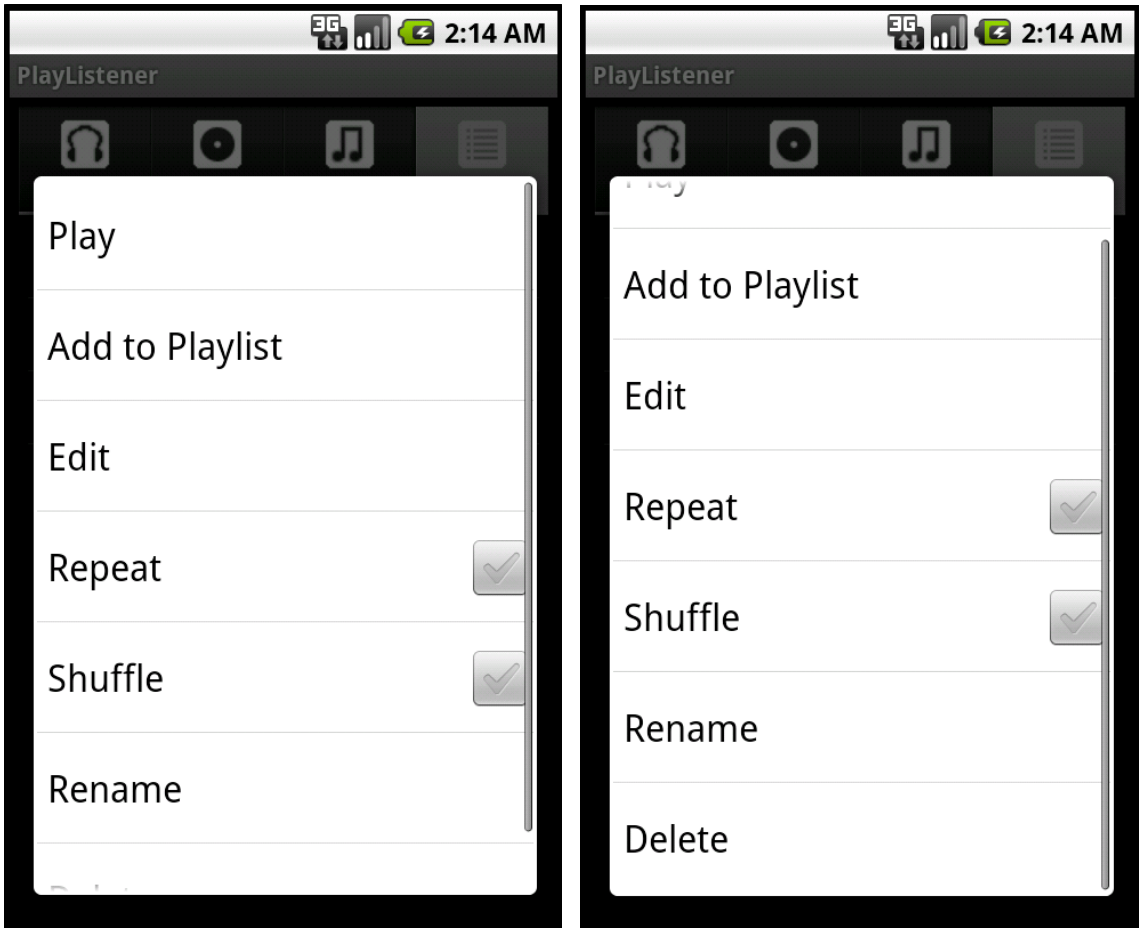

The songs tab is simply an alphabetized list of every mp3 song stored within the device. The playlist tab is similar except the list of playlists is not alphabetized. It is ordered in chronology of creation. First created is first listed.

The user has the ability to create new playlists. The user can hold down on one of the listed Music Objects. A submenu will appear giving the user a choice to either play the current object and all of its children or add the item to a playlist. When selecting ‘Add to Playlist’, another submenu will pop up. This menu shows the user all previously created playlists in the order that they were created. The user also has the options to either queue an item to the end of what is currently playing, or create a new playlist. If a new playlist is to be created, a pop up box requiring text input for the name of the playlist will appear. Once the name is entered, the ‘Ok’ button can be selected and the playlist is created. The user can then add items to the newly created playlist and should be listed under the Playlist tab section.



When the user holds down to reach the submenu for a playlist, the submenu is different. There are several additional options that the user can select within this scrollable menu. The user can enable the playlist to shuffle the elements it contains and/or repeat, or loop, the playlist, edit elements within the playlist, rename the playlist, or delete it.



**Appendix B: Power Point Slides**

# Hierarchical Playlists

Jeremy Bradshaw
Aaron Baldwin
Kevin Mathew

## History

- Originated from radio stations limiting list of songs to be played
- Refers to order of songs within time period
- Hasn't been changed in decades

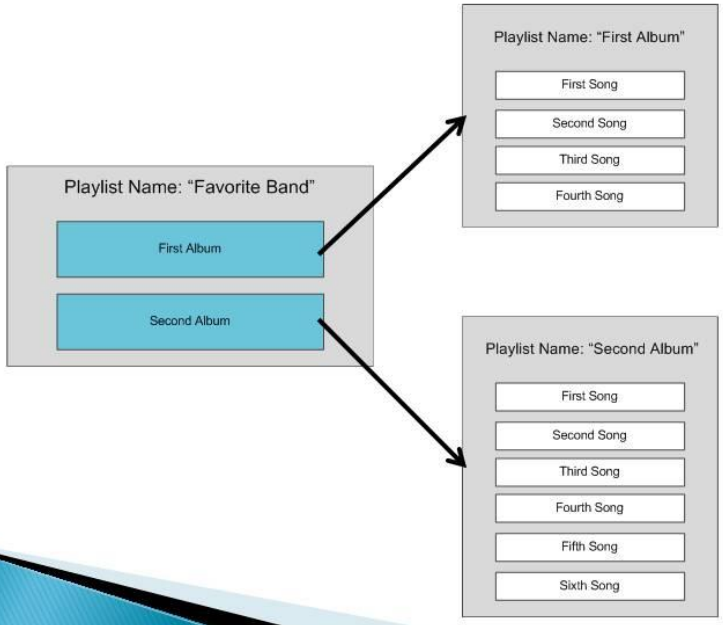
# Problem

- Combining multiple playlists
  - Messy
  - Duplicate Entries
  - Organization Lost
- Core Structure Unchanged

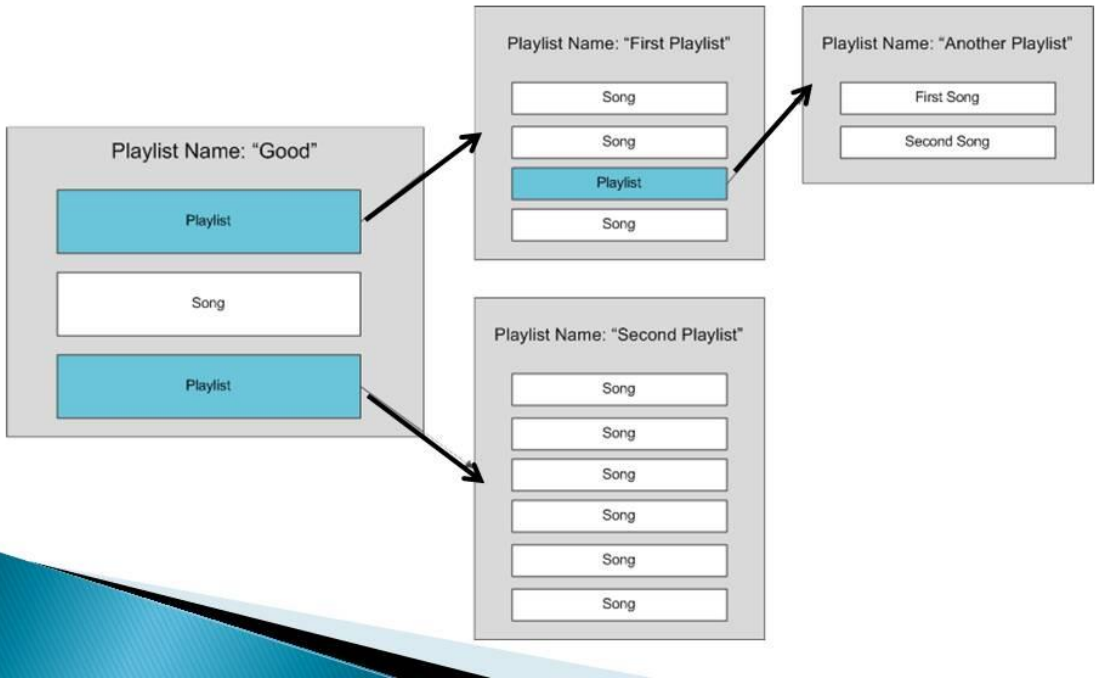
# Our Solution

- Hierarchical Playlists
  - Resembles structure of classic file systems
  - Allows for granular control of playlists
  - More robust structures possible
- Changes fundamental structure
  - Object-Oriented design
    - Model-View-Controller
  - Updated for modern listening patterns

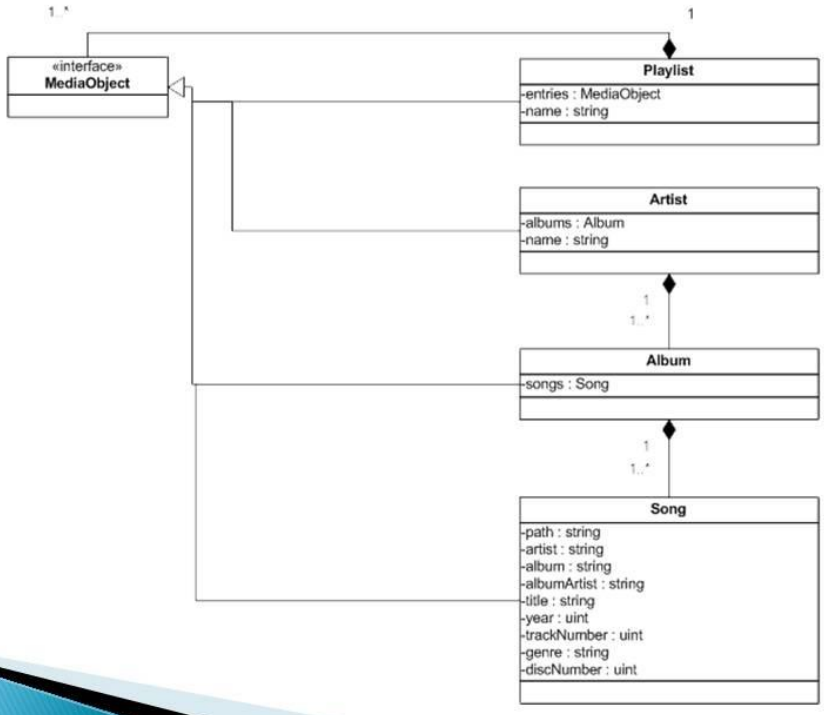
# Example




# Another Example



# UML




# Platform Choice

# Demo

# Q & A

## Appendix C: Code Listings

### Music Object

```
public interface MediaObject extends Serializable {
   public MediaObject get(int index);
   public int size();
   public String toString();
   public MediaObject getParent();
   public void setParent(MediaObject parent);
   public Song getNextSong();
   public Song getPreviousSong();
   public void resetPositions();
}
```

### Playlist

```
public Playlist(Playlist oldList) {
      position = 0;
      this.repeat = oldList.repeat;
      this.shuffle = oldList.shuffle;
      this.parent = oldList.parent;
      this.name = oldList.name;
      entries = new ArrayList<MediaObject>();

      for(int i = 0; i < oldList.size(); i++) {
         if(oldList.get(i) instanceof Playlist) {
                  entries.add(new Playlist((Playlist)oldList.get(i)));
         }
         else if(oldList.get(i) instanceof Artist) {
                  entries.add(new Artist((Artist)oldList.get(i)));
         }
         else if(oldList.get(i) instanceof Album) {
                  entries.add(new Album((Album)oldList.get(i)));
         }
         else if(oldList.get(i) instanceof Song) {
                  entries.add(new Song((Song)oldList.get(i)));
         }
      }
   }
```

//Seriously considering doing javadocs and uploading them to a website instead of code snippets