CSCI 4237 Final Project Report

SoundCloud Utility

December 10th, 2012
Steven Landis

Introduction

For my final project I created a SoundCloud Utility application. This app allows users to record and upload sounds on the go, view and listen to their own tracks, and also view and listen to tracks uploaded by other artists.

Implementation

The application utilizes a number of frameworks including: AVFoundation, Security, QuartzCore, CoreText, CoreLocation, CoreGraphics, AddressBook, and AddressBookUI. Most of these frameworks besides AVFoundation were needed to be able to successfully compile code included in the SDK. Although I did not personally take advantage of these frameworks, there are many dependencies which needed to be satisfied so I that I could use the functions pertinent to my application. However, I did become very familiar with the AVFoundation framework and its audio playback and recording capabilities.

On the main screen, there are 4 buttons: Login, Record, Upload, and Tracks. When the Login button is pressed, a manual segue is performed to a login view-controller provided by the SoundCloud SDK. This allows users to login and authenticate my application to make various calls with their account once logged in. From the main view controller, a user can also choose to record a song. After songs are recorded, they are stored locally in the program data; each recording can be seen in the dynamic table view. Once a user touches a cell representing a corresponding recording, the recording will play. The last cell selected can be uploaded to a user's SoundCloud tracks by pressing the Upload button. The uploading is primarily handled using code from the SDK.

On the main view-controller there is a text box and a Tracks button. If a user presses on the Tracks button without entering in any text, a segue is performed and their own uploaded tracks will be displayed in a dynamic table view. If the username of an artist is entered into the text box and the Tracks button is pressed, the tracks uploaded by the corresponding artist will be displayed instead. A user then has the option to press a certain cell. At this time, a segue is performed to the track display view-controller. This view provides the following information and data: track artwork, track waveform, artist, and track title. The play button on this view-controller is disabled until the track is ready to be played. A UIIndicatorView spins at the navigation bar during buffering to let the user know meaningful progress is being made.

All information about tracks is pulled from SoundCloud using their web based API. SoundCloud's API allows developers to take advantage of JSON objects which are parsed

using NSJSONSerialization objects. This object returns arrays, strings, and dictionaries which are then used to retrieve specific information about the tracks to load for a certain user, as well as the data to load for a specific track.

Problems

It took quite some time for me to figure out how to have a track playing in the background so that a user can go to the home screen and still hear the track they selected. In addition, because I was initializing my AVAudioPlayer in the last view-controller, I ran into issues where when going back to another view-controller the track would stop playing. I determined that this was because the view-controller was popped from the stack and the objects associated with it were deallocated. I realized that the solution to this was to have the AVAudioPlayer be a property of the AppDelegate. By doing this, no matter which view-controller I was in, I was able to continue playing the track selected as well as change the delegate of the player to the current view-controller. This allowed me to be able to perform any necessary delegate methods regardless of view-controller by setting the delegate in the viewWillAppear method.

Another problem I had was figuring out how to successfully record sounds and store them locally. It turns out that I needed to set the category of the sharedInstance of type AVAudioSession to AVAudioSessionCategoryRecord. A smaller issue related to recording that I encountered was figuring out how to save the recordings as a file type supported by SoundCloud; I ended up going with .aac which turned out to work quite well.

In general, I ran into quite a few issues where there were bad memory accesses and exceptions were thrown due to certain information being missing. For example, when I uploaded recordings through the app without setting a picture for the artwork and then tried to play the track, the app crashed because it was attempted to access the URL corresponding to the artwork which did not exist. As such, I added additional code to avoid bad memory accesses in cases where some information was not available.

Finally, because the data that is fetched from SoundCloud happens in an asynchronous thread, I had to be very careful about what I was doing in the thread and keep in mind multiple things were happening at once. A specific issue I ran into was when I had my spinning indicator in the thread and even though I used the stop method in the thread, it continued to spin regardless.

Lessons Learned

After finishing this application, I learned that it is very important to understand how specific SDKs and APIs work if you are going to use them. If you are able to understand them well, they are incredibly valuable and can help you build some amazing applications. It is also crucial that as a developer you understand the classes you are working with so that you do not forget to implement any necessary methods or properties that must be set.