Steve Geluso **/geluso@(cs|uw)/**
Aryan Naraghi **/aryan(@cs|90@uw)/**

30 March 2011

CSE 403 Project Proposal: Song Tagging System for the KEXP Radio Station

Vision

Radio listeners often hear stations play songs they love without knowing what song is actually playing. We envision creating a simple application dedicated to allowing Seattle's KEXP radio listeners to identify and keep track of songs that they like. The application will manifest itself as a web app with an Android companion. The web app will be fully featured: users may see what's currently playing on the radio, tag songs they like, manage their favorite songs, view song suggestions, and export their data to other music services (e.g., syncing favorites as a playlist to Grooveshark). The Android app will allow users on the go to easily tag songs that they like and access a list of their tagged songs. User data between the web app and the mobile app will always be consistent. The data for what song is playing at a given time will be drawn from KEXP's website. Similarly, the suggestions feature will be based on the 10+ years of playlist history available on KEXP's website.

The Android app will primarily allow listeners to tag songs. When listeners hear a song that they like, they can take out their Android phone and press a button to have the application remember the song that is playing on KEXP at the time. This eliminates the need to memorize or write down parts of the lyrics (a dangerous activity for drivers) in hopes of looking up the song later. The application would also work without an Internet connection by recording the time when the button is pressed and querying the server later to find out what song was playing at the time. This is a compelling use-case for many radio listeners who listen to the radio while driving.

We may face competition from other developers making their own mobile apps. There is currently an iPhone app that streams KEXP to listeners and allows them to tag their favorite songs. The iPhone app is not coupled with a website; it does not integrate any third party music services nor does it have a song suggestion feature. Our only direct Android competition comes from an app KEXP promised to deliver in fall of 2010. No app was ever released. This leads us to believe that the project is dead. We want to be the first dedicated KEXP app in the Android market, and we want our app to be better than the iPhone app.

Software Architecture

The software will consist of three main components: a web crawler, a database and server, and the web and mobile applications. **Figure 1** is a simple overview of the system.

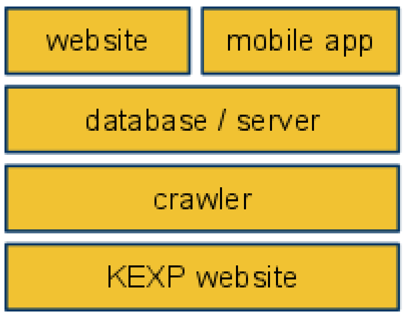

Figure 1. Each level of components is dependent on the levels below it.

The crawler will have two purposes: to regularly check KEXP's website for the song that is playing and to collect the 10+ years of playlist history that is available. The former task is accomplished by accessing an XML file that KEXP publishes, indicating what song is being played. **Figure 2** shows what this XML file looks like. To maintain up-to-date information, this file will have to be accessed every few seconds. The latter task will be accomplished in a one-time crawl, with the possibility of weekly crawls to add new data.

```
<t>
    <s>Flutter</s>              <!-- song name    -->
    <a>Dial 'M' for Monkey</a>  <!-- album name   -->
    <r>Bonobo</r>               <!-- artist name  -->
    <l>Ninja Tune</l>           <!-- record label -->
    <t>5:17AM</t>               <!-- time played  -->
    <ry>2003</ry>               <!-- release year -->
</t>
```

Figure 2. A sample of what KEXP's playlist data feed looks like.

We intend to use Django (a Python Web Framework) for our backend and the web application. Django will tie in a database (most likely MySQL) for use with the web application and the Android app. The Android app will submit queries to the database asking "what song was playing at *this* time?" As such, a crucial role of the database and server will be accepting timestamps and returning song information. The database will also store data that will allow us to make song suggestions.

We plan to use historical playlist data to generate music suggestions. We will employ an algorithm that extracts N-grams from a text document. These N-grams represent what words appear near each other within a document, indicating that they may be semantically related. We will create a document representing the last ten years of music played on KEXP. Each song will be given a unique ID, that will be tokenized as a word. We can generate N-grams based on the proximity of these song ID's to find out which songs have been played near each other. Every song will have its own N-gram and that relates it to a set of other similar songs. This feature can be used to generate playlists (e.g., iTunes Genius, Pandora), or to recommend a small set of similar songs, (e.g., Amazon's "if you like this, then you might also like these").

We intend to serve everything on either Google App Engine or Amazon Web Services. Relying on cloud-based services will allow us to worry less about hardware and will also make the service more scalable.

The Android app will be written in Java and will depend on the Django server for data. Information between the server and the app will be delivered as JSON-serialized data, which will make serializing and parsing of data easy. The phone will be able to keep a local copy of user information, such as the user's favorite songs. As the phone queries the server for song information the server will synchronize data from the phone with the web application. Users will always have the same data available on both the web app and on their phone.

Challenges and Risks

Since KEXP does not publicly advertise its data feed (the XML file depicted in **Figure 2**), we will be crawling it on our own. KEXP's robot.txt allows robots to crawl this file. We plan to abide by typical robot rules and only issue one request every six seconds. Unfortunately, since we can only make at most one request every six seconds, our data may not be accurate between the end of one song and the beginning of another. To combat this, we will need to provide a mechanism allowing users to correct misidentified songs.

It will be challenging to ensure data is consistent between the web app and phone app as user data is distributed over these different mediums. If users are allowed to modify their data in both locations we will need to ensure that data is not corrupted when the two parts try to synchronize with one another.

We are not sure how well the song suggestion feature will work. Pursuing this feature is risky because it may take a lot of time to perfect, or it may not even work at all. We intend on mediating this risk by performing some experimentation with the idea before going forward with it.