CMSC 324

Patrick Collins, Zachary Jenkins & Mark Landgrebe

May 4, 2015

1 Contact Info

Team Name: PlayListr

Member	Email
Patrick Collins	pscollins@uchicago.edu
Zachary Jenkins	zjenkins@uchicago.edu
Mark Landgrebe	mlandgrebe@uchicago.edu

2 Motivation

3 Background

4 Project Specification

The fundamental purpose of PlayListr is to improve the experience of guests at events playing music. However, there are a variety of avenues by which we can achieve this goal, and we hope to explore as many of them as we can under the time constraints.

4.1 Capabilities

We can divide our planned features in to a hierarchy, such that each higher level depends upon a correct implementation of the previous one:

1. Core functionality.

   The core functionality of the app is to allow a host to create a "room," and to allow guests who join the room to have control over the music that is played.

   However, a variety of structure must be in place in order to accomplish this goal. The core functionality is the most ambitious part of this project — a successful implementation requires:

   - Correct interoperation with Spotify in order to authenticate users and stream music.
   - Construction of a database to hold identifying information about users, with sufficient information to reconstruct the user's Spotify account details, as well as the details of all existing rooms and the songs enqueued within them.
   - Development of a server that exposes a REST API, giving access to the database and allowing simple CRUD operations.
   - Ensuring consistency across many mobile users connected to the same room, taking in to account the possibility of high network latency over cellular data networks.

   With this in mind, we have focused our current development efforts on building the core functionality before tackling the features that follow — in the "software architecture" section below, we outline an implementation of the core functionality.

2. Location awareness.

   Users should be prompted to join a room whenever they come within a pre-set distance of an existing room via push notification.

3. Smartwatch integration.

   Users should be able to vote on songs and see the currently playing song from the display of their Android smartwatches. This will likely take place via Bluetooth communication with their smartphones. This step is crucial for the following step since smartwatch sensor data will allow for a much more accurate picture of user activity within a room.

4. Adaptive song selection on the basis of sensor data.

   The app should detect when the activity level within a room rises or falls and play faster- or slower-paced music, accordingly. For simplicity, we will simply select songs currently in the queue and pad tempo-matched songs with artificial votes in order to float them to the top of the queue — this avoids problems of trying to match the genre of the music currently being played.

   We can take a variety of angles in processing sensor data, but the simplest is likely to be taking the average norm of the acceleration vector of each user per unit time. For example, given that we have accelerometer data for the user's acceleration $\vec{a}$ such that:

   $$\vec{a} = (x, y, z)$$

   And, letting $\vec{a}_u$ represent the acceleration vector for the user u belonging to the group of users U in a room during the previous tick, we define the activity level l of that tick to be:

   $$\begin{aligned} l &= \sum_{u \in U} \frac{||\vec{a}||}{|U|} \\ &= \sum_{u \in U} \frac{\sqrt{x^2 + y^2 + z^2}}{|U|} \end{aligned}$$

   Yielding a natural and intuitive notion of "activity level" that can be quickly computed.

5. Social features.

   Users have a "friend list," consisting of other users, that allows them to initiate communication (perhaps via a builtin messaging service, or a tie in to an external messaging service, such as Facebook Messenger).

6. Gesture recognition.

   Users can add each other to their friend lists by shaking hands. Time permitting, we can explore other gestures, such as a high five to indicate that both users upvote the current song.

7. User pairing on the basis of activity level.

   Given that the server will have access to the activity levels of all users in a room, users can opt-in to a "friendly mode" in which they will be periodically asked if they would like to become friends with another user in the room based on similar activity levels (and perhaps similar music preferences). If both users accept, then a conversation is initiated between the two.

4.2 Software Architecture & Design

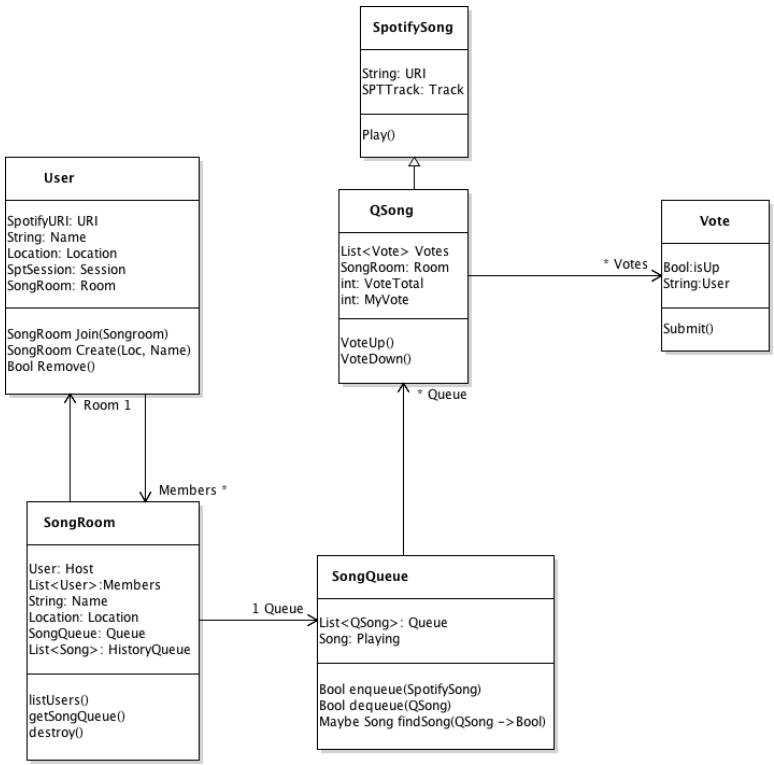

Figure 1: UML Diagram of Client-Side Code

5 Planned Demonstration

Figure 2: UI Mockup

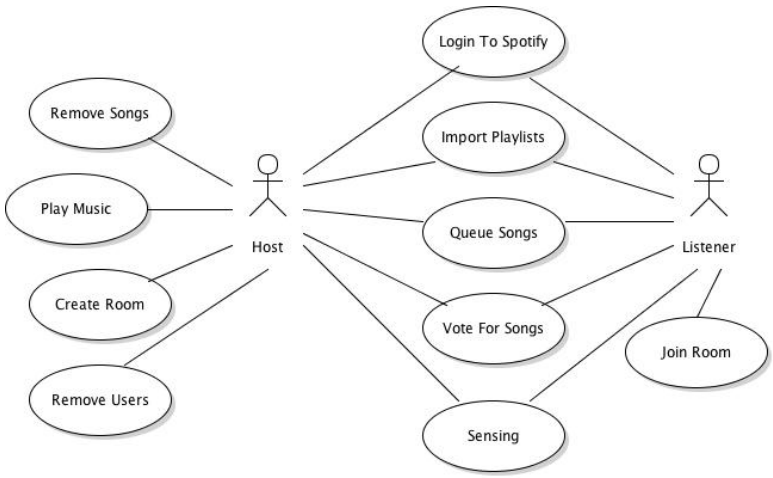

Figure 3: Use Case Diagram

6 Implementation Schedule & Supporting Tools