Pitchslapp

COS 333 Spring 2016 - Prof. Moretti
March 15, 2016

Caroline Congdon - ccongdon@princeton.edu
Andrew Hunt - amhunt@princeton.edu
Zachary Stecker - zstecker@princeton.edu
Team Leader: Andrew Hunt

Overview

The primary purpose of Pitchslapp is to provide collegiate a cappella groups with a comprehensive and centralized system for managing their musical repertoires. The app will be used by singers in a cappella groups, and the app experience will be consistent between members of the same group. We are designing with the experience of Princeton a cappella groups in mind, but groups on any campus will be able to benefit from this app.

Two of our group members participate in a cappella at Princeton, and we have noticed a number of persistent problems. A single group may be responsible for upwards of 40 songs at any given time, each of which carries a significant amount of data: starting pitch (key), soloists, sheet music for the arrangement, song length, active/non-active in current repertoire, mood, and more. Anytime the group's director wants to put together a setlist for a performance, s/he has to consider all of this information, which tends to be scattered across forgotten pieces of paper, old books, and iPhone notes. Our app will consolidate these songs and the data each one carries, allowing for easy searching, setlist creation, and pitch retrieval.

Our goal is to build versions of this app on two platforms: web and iOS. Both will implement all of our central features, but each will have its own focus. Our web interface will prioritize data entry, allowing users to quickly and easily enter information for the songs in their repertoire. Our iOS app will prioritize setlist creation and playing starting pitches. Both will focus on displaying sheet music, searching, and editing song details. Primary views and data structures for the user are (1) a central, searchable list consisting of all songs in the repertoire, and (2) a series of setlists that can be designed for different performances.

We believe that this app will become an indispensable tool for a cappella groups. A user will be able to build and modify a setlist on the go (allowing other users in the same group to view the changes) and use that setlist during a performance to easily play starting pitches. Our search function will allow the user to find that perfect missing song based on soloist, mood, or length. Our sheet music viewer will allow all group members to quickly review a measure or two during rehearsal or before a performance on the go, settling any vicious aca-disputes in mere seconds.

We will build this app using Swift for iOS and JavaScript for web, with Firebase as our central database.

Requirements and Target Audiences

Currently, many of Princeton's a cappella groups have confusing, disjointed, and problematic systems set up to store their music and plan their set lists. When a group has an arch performance coming up, they may have to look through various emails, spreadsheets and repertoire documents to come up with a set list. In addition, finding PDFs of sheet music is often difficult and decentralized. Our target audience will be college a cappella groups and other musical groups that have a repertoire. This app is designed to be highly adjustable depending on user preferences, so there are many use cases.

Functionality

The overarching functionality of Pitchslapp is to make the process of organizing and interacting with musical data easier for all member of a capella groups (originally, Princeton groups, but eventually groups beyond this campus). Below are some scenarios in which tasks associated with a capella are made significantly easier via the use of Pitchslapp.

Scenario 1:
The music director of the Footnotes wants to create a setlist for an upcoming "holiday"-themed arch. He isn't really sure what holiday songs the group has available off the top of his head. He also wants to add some variety and is interested in finding songs that the group hasn't practiced in a long time. He pulls up the Pitchslapp website, logs in, and is able to search the Footnotes' entire repertoire for songs tagged "holiday" that could work for the arch. He avoids the unwieldy and time-consuming task of sifting through tons of randomly distributed spreadsheets, emails, and paperwork. When he finds a song he likes, he can add it with one click to his new setlist. Later that night at rehearsal, members of the group can view the setlist that the musical director made on the Pitchslapp app on their phones. They can look over the sheet music, listen to a recording of a song from an album released 6 years ago, and even use the app to play the starting pitch for each song.

Scenario 2:
A member of R20 misses rehearsal one night because he was too busy working on a computer science project. Later, he wants to do a little bit of practice on his own of a song that he knows R20 rehearsed tonight. He logs in to Pitchslapp on his phone and finds the song by searching the repertoire page. He opens the sheet music for his part and scans it to jog his memory before practicing.

Scenario 3:
A new member of the Tigerlilies has an idea for a new song to add to the group's repertoire. Before texting the music director, she quickly searches the song on Pitchslapp to make sure it isn't already there. She sees that it's not, and suggests it to the music director, who thinks it is a great idea. After checking with the appropriate channels in Acaprez, the music director is able to add the song to the Lils' repertoire. She opens Pitchslapp on her computer and clicks to add a new song to the repertoire page. She is able to add sheet music, choose the appropriate starting pitch, enter the length of the song, as well as adding some tags to help identify it via search. The new member who originally suggested the song can now check Pitchslapp on her phone and see the song (and its accompanying data) whenever she wants. Without the use of Pitchslapp, this entire process would have resulted in the song being written down in some random Google Doc or Excel Spreadsheet, the sheet music being saved in a DropBox that not all members of the group have access to, and the rest of the group wouldn't necessarily even be aware that the song was available to the group.

Design

Data Management:
We are planning on using Firebase as our cloud storage solution. Firebase has an easy to use API for iOS and web development and one of our team members has experience using it, so it seems like the best option for our multi-platform application. It also enables real-time syncing between devices without the need for clicking a refresh button, which is helpful when multiple people are trying to edit the same dataset.

User Interface:
Our main user-facing application will be an iOS app, developed using Swift/Xcode, that contains the vast majority of the functionality AND the app interface. The app will consist of two main views. First, the repertoire view will show all of the songs in the group's repertoire and allow users to examine various parts of the song, add supporting files, etc. (as described in the functionality section). There will be a search bar at the top as well to quickly find a song in the groups rep or to search another field (e.g. search songs that have "sad" in a user-made "song mood" field) Second, the set-list maker view will allow users to pull in songs from the repertoire for a set list, switch around and arrange the list, and play starting pitches for the songs in the current set. Both of these views are pretty simple to create with Xcode (templates come very close to our end goal) so most of the work will be in making the forms intuitive and beautiful.

Our website will be similar to the app in functionality and views, with a couple of other features, like importing multiple repertoire songs at once and being able to upload PDFs of sheet music. We're not sure what web frameworks we'll be using for the website yet, but the interface between

the functionality (process tier) and the user interface will likely just be the Firebase API JS functions.

Process:
The functionality will be created for the mobile app and the website using Swift and JavaScript, respectively. The core functionality of our app is mostly user interaction with the information in the cloud (on Firebase). Thus, the process layer is actually pretty small because it will simply connect our backend of Firebase to our frontend of JavaScript and Swift. Because the Firebase API does this for us (in both JavaScript *and* Swift) with easy functions for retrieving, storing, and changing information in the cloud server, our process tier is simply calling these functions correctly to store and manage the repertoire information for every group that uses the app.

In addition, we will need to store usernames, passwords, group identifiers, permissions, etc. We're not sure what authentication system we're going to use, but since we only need basic authentication functionality, there are multiple options. Firebase does provide an authentication service, and it may make sense to use this since we are using Firebase to power our database. We will need to consider a way for multiple members of one group to authenticate as members of that group.

Timeline (course requirements are in **bold)**

Deadline	**Milestones**
3/21	● **Project status website up and running**
3/28	● **Prototype:** display primitive list of dummy songs from database on iOS ● Firebase is initialized and parameters are set for each song's data ● Web displays basic list of songs from Firebase
4/3	● Build capabilities to add and edit songs in the database through iOS and web apps, with changes reflected in central repertoire lists on both platforms
4/10	● Database: ○ build setlist functionality into the database ○ implement a way of storing sheet music for each song, either by linking through Dropbox API or otherwise ● iOS & Web: implement setlists, allowing user to view, rearrange, add to and remove from setlists continued...

4/18	● **Alpha version** ○ website and app display central list as well as setlists ○ user can add, edit, and remove songs and setlists ○ user can view sheet music for songs
4/25	● **Beta version:** cleaned up alpha version, plus: ○ central list can be searched and filtered ○ can play starting pitch for a selected song
5/4	● **Ready for demo**
Dean's Date	● **Final submission**

Risks and Outcomes

There are a few risks associated with the platforms we have chosen to work with. An iOS app is a major element of our project, and only two of our team members have Macs, which means that we don't have the maximum convenience level when working on the app. Similarly, none of the three of us are experts in the various technologies we have chosen, so there will likely be a bit of a learning curve which could take more time than we expect, depending on how complicated certain features end up being. With only 3 team members, we will have to make sure to budget our time accordingly if we want to complete both the mobile app and web app with all of our intended features. In case of any issues achieving this goal, we have chosen the features of our app specifically so that a baseline can still be functional without every single feature that we have planned. If we can make it to the baseline, it will simply be a matter of adding whatever we have time to add before the end of the semester. No one feature will make or break our project. Another problem we anticipated is the lack of test data or a group that is willing to beta test our project. Luckily, two of our group members are involved in a cappella groups on campus that would be willing to provide data and help to test both the web and mobile platforms so that we can be sure to vet the usability and value of our app before the end of the semester.