Jukebox (formerly SharedMusic)

For the code review, I have decided to to focus on the socket.io portion of our app (namely, the ./app.js file) . I chose this code for a couple of reasons. In particular, the socket.io code seemed like a dense section in our codebase. It's the heart of our app and important as it does all of the backend work of handling events.

As I was walking through it, it was interesting to see how all of the architecture (rooms, users, songs, etc) gets woven together with the socket events such as joining a room, adding a song, etc.. At first glance, I could tell there was a few spots in which some copying/pasting appeared to be occurring. On each socket event, there were checks for errors with the user and room. So the first change I made was refactoring this into its own function, and make calls to that function instead of repeated code. Also, when an error was being emitted it was simply emitting a static string of "User does not exist for userID", instead of emitting the actual userID for more context to the error.

Because the socket.io code heavily uses the architecture.js file, I first had to do some research and studying since I was unfamiliar with OOP in javascipt (heres one of the links I used along with mozilla documentation http://javascriptissexy.com/oop-in-javascript-what-you-need-to-know/). This was helpful and something that I learned from this review! At first glance, the readability seemed to suffer from the extensive use of the keyword 'this' and underscores. It just didn't "feel" like javascript to me. But I looked into some of the libraries we are using such as simplesets.js and appears to appears to be common industry practice. So once I started fixing things, I started with the comments and filled in comments for functions that were missing one. After that I removed a lingering function that wasn't ever being called. I then noticed the line 'this._state.users.has(user)' and a few more ones similar were being repeated a lot, so I refactored it into a simpler functions such as hasUser(user).

I was happy with the socket and architecture code so I decided to move onto the routes/index.js file. This file is simply for defining routes for the webserver of which files it should serve up on GET/POST/etc requests. There were a lot of individual routes to specific files, namely frontend js and html files. So everytime everyone wanted to add a new html file they were manually adding a route for that html file. Instead, I got rid of all of the individual routes and made it so the webserver will automatically look in the 'views', 'public', 'documentation' folder when GET requests were made. There were also some duplicate routes that were both in app.js and routes/index.js that I got rid of. Routes shouldn't live in app.js, they should be separated out into their own routes/index.js file for clarity and modularity.