Team Number: Section 011 team 4

Team Members:

Bernardo Fortunato

Bo Zhang

Dalbir Brar

Jared Jewell

Jacob Levato

Team Name: Mr. Worldwide

Demo:

Four application features completed:

1. Register Account with 1 email (User enters name, email, password, confirm password)
2. Sign In to Registered Account
3. Profile page HTML/CSS layout
5. Password/ConfirmPassword must match & password is encrypted when stored
6. Home Feed greets user with “Welcome <user’s name>”
7. Home Feed HTML/CSS layout

Application features that worked:

1. Register Account with 1 email (User enters name, email, password, confirm password)
2. Sign In to Registered Account
3. Profile page HTML/CSS layout
5. Password/ConfirmPassword must match & password is encrypted when stored
6. Home Feed greets user with “Welcome <user’s name>”
7. Home Feed HTML/CSS layout

Suggestions:

Look at the Spotify API, document what endpoints are available to us.

Tracks/Songs/Artists/Albums

Spotify URIs and IDs

In requests to the Web API and responses from it, you will frequently encounter the following parameters:

PARAMETER	DESCRIPTION	EXAMPLE
Spotify URI	The resource identifier that you can enter, for example, in the Spotify Desktop client's search box to locate an artist, album, or track. To find a Spotify URI simply right-click (on Windows) or Ctrl-Click (on a Mac) on the artist's or album's or track's name.	`spotify:track:6rqhFgbbKwnb9MLmUQDhG6`
Spotify ID	The base-62 identifier that you can find at the end of the Spotify URI (see above) for an artist, track, album, playlist, etc. Unlike a Spotify URI, a Spotify ID does not clearly identify the type of resource; that information is provided elsewhere in the call.	`6rqhFgbbKwnb9MLmUQDhG6`
Spotify category ID	The unique string identifying the Spotify category.	`party`
Spotify user ID	The unique string identifying the Spotify user that you can find at the end of the Spotify URI for the user. The ID of the current user can be obtained via the Web API endpoint.	`wizzler`
Spotify URL	An HTML link that opens a track, album, app, playlist or other Spotify resource in a Spotify client (which client is determined by the user's device and account settings at play.spotify.com.	`http://open.spotify.com/track/6rqhFgbbKwnb9MLmUQDhG6`

User-specific endpoints

- Images
  - ugc-image-upload
- Listening History
  - user-read-recently-played
  - user-top-read
  - user-read-playback-position
- Spotify Connect
  - user-read-playback-state
  - user-modify-playback-state
  - user-read-currently-playing
- Playback
  - app-remote-control
  - streaming
- Playlists
  - playlist-modify-public
  - playlist-modify-private
  - playlist-read-private
  - playlist-read-collaborative
- Follow
  - user-follow-modify
  - user-follow-read
- Library
  - user-library-modify
  - user-library-read
- Users
  - user-read-email
  - user-read-private

Individual contributions:

1. **Jake Levato -** I completed the Registration and Sign In mechanism using Nodejs and MySQL. I set up our sign-in/sign-up page to insert and pull user information from our user database. I modified the sign-in/sign-up page to fit the theme of the website along with redirecting to the home page and displaying “Welcome <user’s name>” after being fetched from the database.
2. **Bernardo Fortunato** - I completed the front-end of the sign-in/up page for the service and made it look pretty.
3. **Dalbir Brar** - I updated the profile page to include many tools in the navigation bar. I established a color theme and readable text for the main fields of music to be displayed. I used grids to organize the user’s data (username, description, friends list) to be easily

accessed. I added the queue object to the profile page for the user to see what songs they got sent from their friends.

4. **Jared Jewell** - I built the rudimentary profile page that Dalbir finalized, and built/stylized the currently-working home page, both responsive using flexbox & grid.
5. **Bo Zhang & Jared Jewell** - Created the architecture diagram
6. **Bo Zhang** - Caught up on the project. Contributed to the architecture diagram. Working on the explore html page this week.

Issues Faced

- No notable issues as far as team dynamics, workload, etc.
- Points that we need to clarify on internally
    - What form should recommended songs take for a user? I.e. where should they live?
    - What should queueing look like? Should a user have a single, global queue that all of their friends can add to, or should each friend have a queue for that user that is private?
    - Hosting Database/Node/Express server

Architecture Diagram