Radio Beta Release

Hi all,

I've finished my assessment of the beta radio release. I've captured my notes under a few broad headings:

1. Primary functional tests
2. Exploratory tests
3. Bugs
4. Notes for Discussion
5. Code Review
6. Summary and Next Steps

My test session was conducted on:

- OSX `10.10.5` and `10.9.5`
- Application release version: `1.0.23.90.g42187855`
- Radio release SHA: `cb2a761`

The summary of my findings is that Radio is ready for release. Whilst some minor issues were found, none of them are significant enough in my view to prevent getting this out there and letting us get more data about how it's used.

Primary functional tests conducted

These were the initial set of tests conducted when checking the core functions.These should be the primary regression tests, either checked manually or preferably through an automated test.

- I can create a radio station for an artist - **PASS**
- I can create a radio station for a song - **PASS**
- I can create a radio station for a genre - **PASS**
- I can search for a new radio station to create - **PASS**
- Starting a radio station creates the radio queue - **PASS**
- I can thumbs up the currently playing song - **PASS**
- I can thumbs down the currently playing song - **PASS**
- Creating a station saves it to my stations - **PASS**

Exploratory tests conducted

These were other checks done as part of an exploratory session. If you were to automate these tests, these could become unit tests or integration tests.

- Shuffle should be disabled - **PASS**
- Queue (Next Tracks) should be filled with songs from radio when radio is started - **PASS**

- Manually queued songs should trump the Radio queue and appear as Next Tracks - **PASS**
- I can start a Radio
  - From song (right-click) - **PASS**
  - From artist (right-click) - **PASS**
  - From artist (page, dropdown menu) - **PASS**
  - From radio page - **PASS**
  - From playlist
    - Manually created playlist - **PASS** See notes on collaborative playlists.
    - Spotify created playlist - **PASS**
- An empty playlist should not create a radio - **PASS**
- Essential functions should not be disrupted by starting/switching to a radio
  - Play controls - **PASS**
  - System play controls - **PASS**
  - Private sessions - **PASS**
- Integrations should still work
  - Lyrics / MusicMatch - **PASS**
- The currently playing Radio station should be indicated in Your Stations - **PASS**
- When I thumbs up a song
  - it should lead to a regenerated radio queue - **PASS**
  - it should submit the track as a thumbs up and add it to Liked from Radio - **PASS**
- When I thumbs down a song
  - it should lead to a regenerated radio queue - **PASS**
  - it should submit the track as a thumbs down - **PASS**
- Clicking current song should take me to the Radio and the current station - **PASS**
- After quitting mid-stream, I should be able to return to the song - **PASS**
- Playing a radio station should display it to Connect devices - **PASS**

Bugs

These were bugs indentified during the session. Where possible, I've tried to hunt down a root-cause for the bug which I hope helps.

- Genre Radio Station buttons do not render as links on mouseover
  - My guess is this is somewhere in GLUE to work it out. Potentially the button-info class is being overridden. Guess this one really depends on what the style guide calls for in this case.
- After quitting mid-stream, returning to genre radio returns a previously downvoted song.
  - Potentially this is an effect of Genre Radio stations pulling from a

limited set of songs (see Notes)

- After quitting mid-stream, continuing to play song and then clicking through to Radio crashes stream and skips song currently playing
  - Steps followed: start R&B radio, quit on *Usher - Yeah!*, restart let's me continue it, however trying to click through to Radio eg to vote, stops stream and I am jumped to *Chris Brown - Say Goodbye* (after *Jay-Z - 99 Problems*), 2 songs past where I should be.
  - potentially because `player.js` sets `this.currentStation` to null, which I guess means the player is initialized as per the current song in localStorage. Something in there with onStationStarted is probably conflicting here. My guess is this is a scope problem.
  - might be PlayerState not saving/retrieving current station, meaning it's only fetched on the radio page?

Notes for Discussion

These are the sort of things we should talk about these to see if they're to be considered bugs, or just areas for improvement in the future. Some of these might be deliberate choices, however I like to capture these as things that I experienced during exploratory testing.

- Can't modify a vote once cast.
  - If we wanted to modify an existing vote, this would have to send new feedback for storage. This means that the thumbs button state will always be enabled.
- The radio stations list becomes unwieldy at large numbers (eg >20).
- List of radio stations does not wrap around when clicking the next button. Especially problematic for large lists of stations.
- It's not possible to rate past (previously played) songs (as Pandora does).
- Clicking play on album cover after pausing in Radio restarts the song.
- The only way to upvote/downvote a Radio song is on Radio page (secondary clickthrough to Radio page required from currently playing).
- The recommended Radio stations do not regenerate after Radio interaction.
- You cannot search for a Radio station through the normal search bar. The only way to search for potential stations is the New Station button.
- In general, it can feel hard to differentiate between a playlist and Radio at time. For instance, Discover playlists can feel a bit like Radio, particularly if your play history is heavily skewed.
- Liking (upvoting) a song takes a small amount of time to appear in Liked From Radio playlist.
- There is no way to train Radio before a song plays (eg through Next Songs).
- When generating a genre station, it seems as though there is a limited potential set of songs (around 50-100 songs). Pandora treats each genre station as a starting point for exploration, rather than a shuffle feel.

- You can't create a user radio (eg. I like what my friend listens to, I want to explore their songs).
- Manually created collaborative radio stations do not always queue tracks. This was hard to recreate consistently. I didn't consider this a bug at the moment because it seemed highly test data dependent.
- We're not indicating to the friends feed that I'm listening to a radio station.
- Radio stations are unique to a device, meaning that handoff drops some of the features such as voting.
- When visiting Radio as a brand new user, I was suggested one radio station for Drake. This felt a little weird, as you don't know what kind of user I am yet. An empty state encouraging me to create stations with some songs/artists would be better than a artist you have no idea if I like.

Code Comments

I did a super quick code review of the radio and radio-hub repos. Caveat: I'm not as familiar with Angular, so it's a little hard to tell if some things are idiomatic or just spat out as part of the build system.

- `/radio`: Is there a way to not package all of Angular (122K) and jQuery (82K) in?
- Would you want to log specific genre when a genre-station is triggered, rather than just a seed title? Could be my misunderstanding here, perhaps the `spotify:genre` index is enough to retrieve the seed
- radio and radio-hub: Some languages missing translations (maybe not available in those regions?), eg in radio-hub: `hi`, `ko`, `ro`, `ru`, `ta`, `th` all seem to be missing translations.

Summary and Next Steps

In my opinion, Radio is ready for release. The things identified don't break any of the major functions and integration as a whole seems stable.

Future Testing: Progressing from Manual Testing to Automated Testing

In terms of future testing, I think the team will likely move through the following broad stages:

1. Manual QA / Exploratory
2. Automating safety (UI or core functional)
3. Build out automation

1. Release and Immediate Future

The team will probably have to stick with manual testing efforts at the moment until we see how Radio goes.

I would create a checklist of features to verify (the primary functional tests I identified could be a starting point) prior to release. One way for the team to make this part of their workflow is simply for everyone to use the release candidate build as their primary app, and you can only listen to Radio stations (aka dogfooding).

I'd suggest a code freeze 2-3 days prior to the bi-weekly release date and then cut your release candidate at that point. A focused regression and exploratory test session then gives you time to hotfix prior to release.

Another technique you can use is every member of the team rotates through being the "QA" and assessing the product prior to release. It means everyone has to have an appreciation for where the product is at, and that manual QA doesn't become a chore assigned to one person only.

2. Safety

If it looks like we're going to continue with Radio, I would look to build a smoke automated test suite of the primary functional tests. We can probably just write this in Jasmine. This provides safety to ensure we haven't broken anything major.

3. Functional Tests / Test Per Feature

Once future development begins, we can build out this suite with a series of functional tests. Each new feature worked on should come with test(s) covering the happy path as well as any edge cases or requirements.

It might be worth making a very minimal API test suite to check the behaviour and structure of the various services we're integrating with to ensure the test contracts are maintained. This could be in Jasmine, a DSL like Frisby or a different framework.

We might want to consider rewriting the smoke tests into more of a user journey test suite, following the path a user takes across a number of application components and actions, for instance: creating a station, issuing thumbs ups and editing queues. These broader tests should be very minimal, 10 max.

4. Integrations, TDD

Now that we are in the flow of writing tests per feature, we should investigate:

- Following TDD style of preparing tests and developing.
- Refactoring the existing suite wherever possible to move higher-level tests into unit tests for speeds sake. However be careful here if we end up mocking too much away, we might not be capturing the actual behaviour. Consider retaining some of the functional tests.

At this point it's worth considering larger testing concerns such as:

- Actual OS-based testing: How does the overall product behave on OSX and Windows? Can we build or run our smoke tests against real machines?
  - For interest's sake, I hacked together a proof of concept of an OSX automated test using Hammerspoon.
  - Alternative options could be:
    * SikuliX which has the advantage of being cross-platform, as well as letting you code in a wider range of languages (Python, Ruby, Java) as well as an IDE for graphical test creation
    * ChromeDriver for Selenium to drive the app in dev mode on actual OSX and Windows machines
- Performance testing: How does our application behave when the backend servers are under load? Are there rendering limits in the UI?
- Cross-platform testing: investigate if it's possible to send tests to a cross platform test suite such as SauceLabs.

Product ideas for discussion

Some other suggestions for product improvements that I noticed during the test session

- Try different UI designs, potentially even A/B testing them across users
  - Currently it seems like GLUE is pretty tightly coupled to the implementation. Maybe consider ripping player styles out to a different CSS file?
- Better surfacing Radio - it feels like discovery of Radio is pretty hard. If you miss the sidebar item, there's very little to drive you to discovering and using Radio.
- Allow power tweaking/vote editing - This feels like a difference to Pandora. It would be worth testing what the usage of such a feature would be and if this attracts power users more.
- Making Radio stations shareable - this might move Radio to more of it's own object, as in sharing a station could mean sharing the seed track plus the vote history to rebuild the station.
- Indicating friends who are listening to a station - by clicking their station so you can listen along.
- Deeper integration into the echoNest data to get that "woah that's a great song" feeling in there (leading to less of a playlist on shuffle feel, and rather more like listening to a cool DJ)
- Could Radio become a separate app?
  - Can we decouple some of the radio player logic out for a separate Spotify app?
  - Could decoupling logic help other long tail implmentations such as Roku, car, etc?
- Making connect Radio a seamless handoff as we have done in the other player functions. This might require Radio to become a more dominant

part of the schema.

Other Application Notes

These are notes not related to Radio that I found during testing. I'll raise them with the other teams but I've kept them in these test notes as part of the session.

- The Chromium Embedded view still allows zoom in/out on text. It can look ... strange at very zoomed in.
- The MusixMatch app bar wording is poor grammar: "Download now MusixMatch on your mobile"
- There is inconsistent titling of musixmatch/MusixMatch/Musixmatch throughout the application
- MusixMatch link for iOS leads to GB App Store not US.
  - MusixMatch App Store Links:
    * iOS
    * Android
    * Windows
- The About Spotify page copyright tag has not been updated for 2016.
- There is inconsistent naming of the Spotify Support Community (referred to `Help > Spotify Community`) in app.

Generating these notes in different formats

If you'd like to view these notes as a PDF, BUILD.md details how to generate a copy via pandoc.