Release Plan

Product name: Rootify
Team name: Rootify
Release name: Rootify
Release date: T.B.D.

Revision number: 3
Revision date: 10/27/17

High-level goals:

- See your top short-term and long-term Spotify tracks and artists
- See information about selected artists (e.g. associated genres and top songs, etc.) and tracks (e.g. happiness, energy level, key, etc.) and have the ability to preview tracks
- Discover related tracks and artists based on your top tracks and artists
- Generate a recommended playlist based on tracks, artists, and genres that you like
- Visualize your top short-term and long-term Spotify tracks and artists based on several filtering options and also be able to see how your favorite and recommended tracks and artists compare to one another in terms of certain audio features
- Generate a playlist based on tracks that you like

Sprint 1:

- "As a developer, I want to be comfortable with Git and GitHub so that I can keep my team up to date with my progress." (**2** story points)

- "As a developer, I want to be comfortable with node.js, JavaScript, d3.js, jQuery, and the Spotify API so that I can be skilled for later development." (**5**)

- "As a developer, I want to create a Spotify login so that users can login to their Spotify accounts to access the visualization." (**3**)

- "As a Spotify User, I want a sidebar so I can access track filtering options and track/artist details so that I can be able to create a recommended playlist and view information." (**3**)

- "As a Spotify user, I want the ability to switch between long-term and short-term top 5 tracks and artists so I can easily see and discover new music based on my listening history." (**3**)
- "As a Spotify user, I want to find related artists based on who I listen to so that I can discover new artists." (**3**)

Sprint 2:

- "As a developer, I want to be comfortable with JavaScript, d3.js, jQuery, and the Spotify API so that I can be skilled for later development." (**2**; Continued from Sprint 1)

- “As a Spotify user, I want to see related tracks based on what I listen to so that I can discover new tracks.” (**2**)

- “As a Spotify user, I want the ability to select an artist and preview their top songs or preview a single track.” (**3**)

- “As a Spotify user, I want to be able to view artist details (associated genres and popularity) and track details (happiness, energy, tempo, key (tonic), major or minor, etc) so I can learn more about my taste in music and specific artists and tracks.” (**8**)

Sprint 3:

- “As a Spotify user, I want to be able to see a visualize certain features of my top long-term and short-term artists and tracks alongside related artists and recommended tracks so that I can see how each artist and track compares to each other in order to gain better insight into my taste in music.” (**8**)
- “As a Spotify user, I want to have a bar near the bottom to control which audio features of my top tracks and artists (and related tracks and artists) I can visualize so that I can narrow down specific artists and tracks based on criteria and learn about my taste in music.” (**8**)

Sprint 4:

- “As a Spotify user, I want the ability to create a recommended playlist based on my selected artists, tracks, and/or genres and filtering options so that I can explore and discover new music.” (**8**)

Product backlog:

- “As a Spotify user, I want the ability to discover tracks based on one of my top artists or related artists so I can explore more music.” (**5**)
- “As a Spotify user, I want to be able to **visualize** (e.g. pie chart) my top genres based on my top artists and related artists.” (**3**)

Definition of Done:

- Code committed/checked into the GitHub repository
- Code is up to par based on our agreement on code standards
- At least 80% of test coverage
- All unit tests pass

JavaScript Standards (https://www.w3schools.com/js/js_conventions.asp):

Variable names:

- Must use camelCase for identifier names
- All identifies must begin with a letter
- Use meaningful names

Spaces Around Operations:

- Must always put spaces around operations (e.g.: = + - * /) and after a comma
    - var x = y + z;
    - Var values = [“Volvo”, “Audi”, “Lexus”];

Code Indentation:

- Must always use 4 spaces for tab indentation. You can use tabs during development but make sure your editor is configured to 4 spaces for a tab.

Statement Rules:

- Always end a statement with a semicolon
- Put the opening bracket at the end of the first line.
- Use one space before the opening bracket.
- Put the closing bracket on a new line, without leading spaces.
- Do not end a complex statement with a semicolon

Object Rules:

- Place the opening bracket on the same line as the object name.
- Use colon plus one space between each property and its value.
- Use quotes around string values, not around numeric values.
- Do not add a comma after the last property-value pair.
- Place the closing bracket on a new line, without leading spaces.
- Always end an object definition with a semicolon.
- Example:

```
var person = {
      firstName: "John",
      lastName: "Doe",
      age: 50,
      eyeColor: "blue"
};
```

File names:

- Must be all lower case, and no spaces
    - Use underscores (_) instead for spaces
- Keep it simple (e.g. **layouts.js**)

HTML (and CSS) Standards (https://www.w3schools.com/html/html5_syntax.asp):

Use Lower Case Element Names:

- You must use lower case names for all elements
- Good Example:

```
<section>
```

```
    <p>This is a paragraph.</p>
</section>
```

- Bad Example:

```
<Section>
  <p>This is a paragraph.</p>
</SECTION>
```

Close All HTML Elements

- Though HTML will allow this, please always close all HTML elements.

Close Empty HTML Elements

- Always close empty elements:
  - Allowed: `<meta charset="utf-8"/>`
  - Not allowed: `<meta charset="utf-8">`

Use Lowercase Attribute Names:

- Good: `<div class="menu">`
- Bad: `<div CLASS="menu">`

Spaces and Equal Signs

- Good: `<link rel="stylesheet" href="styles.css">`
- Bad: `<link rel = "stylesheet" href = "styles.css">`

Blank Links and Indentation:

- Do not add blank lines without a reason.
- For readability, add blank lines to separate large or logical code blocks.
- For readability, add four spaces of indentation.
- Do not use unnecessary blank lines and indentation.

Style sheets (CSS):

- Use simple syntax for linking to style sheets (the type attribute is not necessary):
  - <link rel="stylesheet" href="styles.css">
- Rules should be written over multiple lines for clarity:

```
body {
        background-color: lightgrey;
        font-family: "Arial Black", Helvetica,
        sans-serif;
        font-size: 16em;
        color: black;
}
```

- Place the opening bracket on the same line as the selector
- Use one space before the opening bracket

- Use two spaces of indentation
- Use semicolon after each property-value pair, including the last
- Only use quotes around values if the value contains spaces
- Place the closing bracket on a new line, without leading spaces
- Avoid lines over 80 characters

Loading JavaScript in HTML:

- `<script src="myscript.js">`

File Names:

- Use lowercase names with no spaces (use underscores to replace spaces)
- Keep the names short and simple

File Extensions:

- HTML files should have a **.html** or **.htm** extension.
    - If the page is rendered with EJS, use **.ejs**
- CSS files should have a **.css** extension.
- JavaScript files should have a **.js** extension.