Spotify Open Recommendation Engine

System and Unit Test Report | Team Spotify Squad

Release: v.1 | 30 November 2021

System Test Scenarios

- **Sprint 1**
  - **User story**: As a user, I want to sign in with Spotify to save recommended tracks to my library.
  - **Scenario**:
    - Sign in:
      - Go to the live website: http://spotifyore.pythonanywhere.com/spotify-open-recommendation-engine
      - Click "Sign in" (white button in the top-right corner)
      - (*You will be redirected to a "Log in to Spotify" page*)
      - Fill in your login credentials and click "Log in" (green button)
      - (*You will be redirected to our web app page*)

- **Sprint 2**
  - **User story**: As a user, I want to generate a playlist with my recommended tracks so I can conveniently access them within my Spotify library.
  - **Scenario**:
    - Generate playlist & save to library:
      - Complete the "Sign in" scenario steps
      - On the live website, click "Recommendations" on the top navbar
      - Select at least one seed genre (checkboxes on left)
      - Click "Generate Playlist" (purple button, bottom)

            - Enter a name for your new playlist (recommendations modal, bottom left)
            - Click “Generate Playlist” (purple button, bottom right)
            - (*You will be redirected to your Spotify library, where your new playlist is saved*)

- **Sprint 3**
    - **User story:** As a user, I want to be able to preview and filter recommendations into a playlist.
    - **Scenario:**
        - Filter recommendations into/out of the generated playlist:
            - Complete the “Generate playlist & save to library” scenario steps
            - *On the recommendations modal (raised after clicking “Generate Playlist” on the Recommendations page)*:
            - The recommendations modal gives a preview of all the recommended tracks - at this point, the playlist is *not* generated nor saved to the user’s library yet.
            - Use the toggles next to each track (green, right side) to select what tracks to include/exclude in the new playlist.
            - Click “Generate Playlist” (purple button, bottom) to create the playlist and save to your Spotify library.
            - (*You will be redirected to your Spotify library, where your new playlist is saved*)
            - Verify that the playlist contains the tracks you explicitly included in the recommendations modal.
- **Sprint 4**
    - **User story:** As a user, I want to be able to search for a song and visualize the search result(s).
    - **Scenario:**
        - Visualize search results:
            - Complete the “Sign in” scenario steps
            - On the live website, click “Search” on the top navbar

        - Enter a query into the search bar (song name, artist name, etc) and click "Search".
        - (*The page will populate with up to 10 song results from Spotify.*)
        - Visually inspect the characteristics of each search result (loaded from the track's metadata).

Unit Testing

- **Test location:**
    - See the **`tests/`** subdirectory on our project repo.

- **Test invocation:**
    - Run **`python -m unittest -v`** at the repository root.
        - The **`unittest`** module will discover and run all test files (**`test_*.py`**) in the test directory (**`tests/`**).
        - The **`-v`** option will tell Python to output the name and result of each test case run.
    - We have continuous integration set up (via Github Actions) to automatically run all unit tests when the workflow is triggered (upon pushing to any branch).

- **Test details:**
    - **`test_search.py`** | Kelly
        - This file contains *11 tests* to cover a range of inputs/outputs to **`search.py`**'s functions, including:
            - valid/invalid inputs to **`validate_and_search()`**, the main endpoint function.
            - valid/invalid inputs to **`search_for()`**, a function called by **`validate_and_search()`**.

- **Invalid inputs include**: missing query parameters, null query parameters, unhelpful query parameters (all whitespace), null requests.
- There are **specific error outputs** corresponding to specific error-causing conditions; if an error is raised because the query parameter ‘q’ was missing from the request query, the error message will say exactly that.
- **All possible error outputs** (known to **`search`**) are exercised by the test suite - even if the conditions causing them are unlikely to happen in the wild.
- The test file utilizes patching and mocks to isolate **`search`** behavior from external dependencies (like the Spotify API and return values from other functions).