COMS W4170: User Interface Design—Fall 2015
Assignment 1: Part 2

Spoti*SPOT*

README

Name: Adnan Firoze

UNI: af2728

Date: Oct 8, 2015

Spoti*SPOT* [Spotify Search App]

I have built a Spotify search app and named it 'SpotiSPOT'. It meets all the desired requirements mentioned in the assignment and my detailed design is discussed below. This document is structured as follows: (i) Technologies and Libraries used (with justification), (ii) Cross-browser behavior, (iii) How to run and (iv) Heuristics considered and analysis, and (v) Best practices used.

(i) Technologies and Libraries used (with justification and dependency):

1. Pure Javascript
2. JQuery
3. HTML5
4. CSS3
5. Twitter Bootstrap (only the CSS file to format the results)
6. HandlebarsJS (for looping through the results)
7. JQueryUI (only to implement the "slider" to input 'year range' – screenshots attached later)

[There are no online dependency except for the actual Spotify API. All required files are attached and submitted.]

(ii) Cross-browser behavior:

My app was tested in the following browsers without any hitch, including audio playback.

1. Google Chrome 45.0.2454.101 (backward tested till 41.0.0.0)
2. Mozilla Firefox 34.0 – 38.0
3. Safari 7, 8, 9
4. Internet Explorer 10 and 11 (it will ask you to enable Javascript unless they are on already at the bottom of the screen)

(iii) How to run:

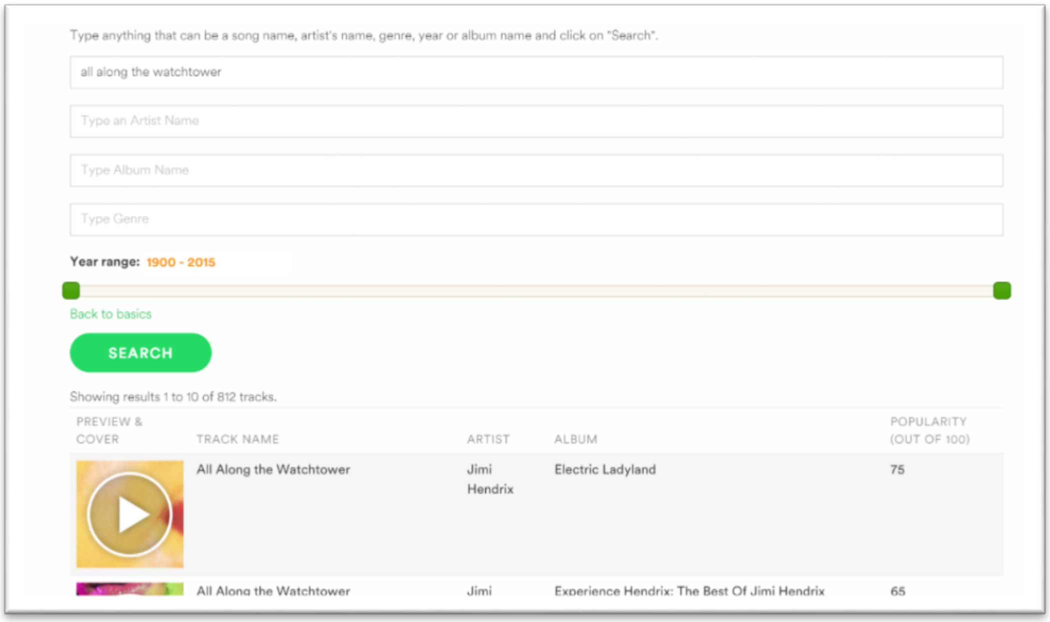
Step 1: Download the folder named HW1 (as a contingency, I have also included a ZIP archive outside it but you will not need that if you download the folder)

Step 2: Simply open "index.html" in your browser (and please make sure you have internet connectivity so you can actually see the results through the API – but the app itself has no online dependency)

Step 3: In the "Advanced Search" mode, you can slide the sliders to define range of year.

Explanation: The 'advanced search' works on an AND logic. Say you search for 'Nirvana' with the year range 1900-present year. You would get 1523 songs (with other artists with the word Nirvana in it like "Approaching Nirvana". Now, if we use the slider to narrow down to 2012-2015, we will get 398 songs.

Another example is the famous song called "All along the watchtower" which was originally by Jimmy Hendrix but many artists performed it later. If we just use the song name, we get the following (812 songs):



Type anything that can be a song name, artist's name, genre, year or album name and click on "Search".

all along the watchtower

Type an Artist Name

Type Album Name

Type Genre

Year range: 1900 - 2015

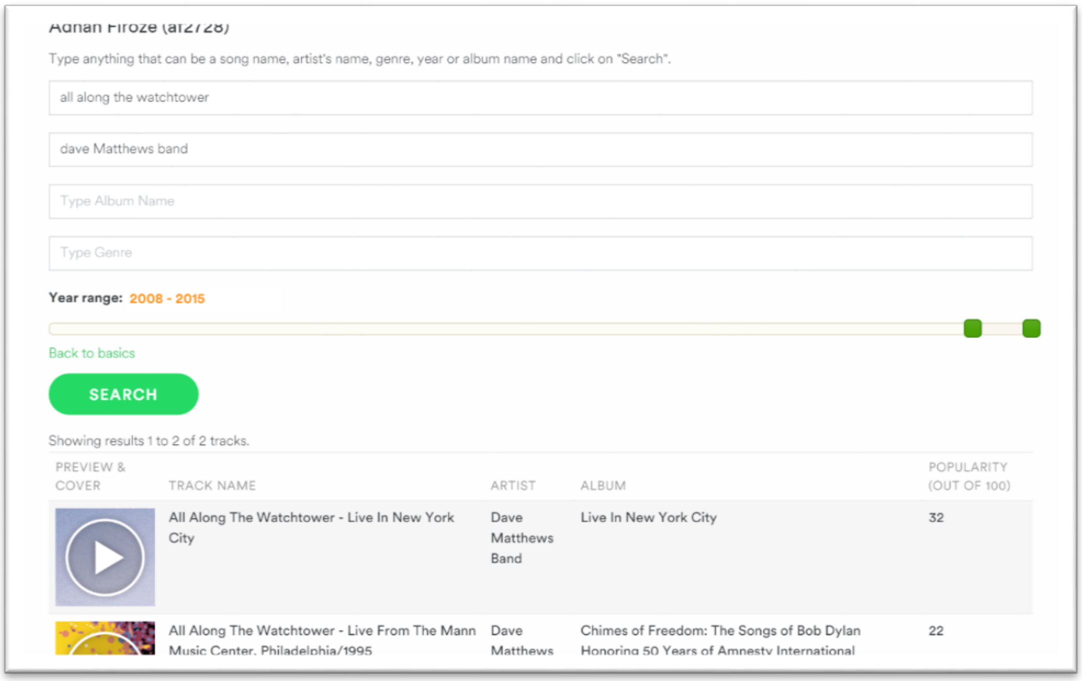
Back to basics

SEARCH

Showing results 1 to 10 of 812 tracks.

PREVIEW & COVER	TRACK NAME	ARTIST	ALBUM	POPULARITY (OUT OF 100)
	All Along the Watchtower	Jimi Hendrix	Electric Ladyland	75
	All Along the Watchtower	Jimi	Experience Hendrix: The Best Of Jimi Hendrix	65

Figure 1: Broad search of 'All along the watchtower'


Adnan Firoze (af2728)

Type anything that can be a song name, artist's name, genre, year or album name and click on "Search".

all along the watchtower

dave Matthews band

Type Album Name

Type Genre

Year range: 2008 - 2015

Back to basics

SEARCH

Showing results 1 to 2 of 2 tracks.

PREVIEW & COVER	TRACK NAME	ARTIST	ALBUM	POPULARITY (OUT OF 100)
	All Along The Watchtower - Live In New York City	Dave Matthews Band	Live In New York City	32
	All Along The Watchtower - Live From The Mann Music Center, Philadelphia/1995	Dave Matthews	Chimes of Freedom: The Songs of Bob Dylan Honoring 50 Years of Amnesty International	22

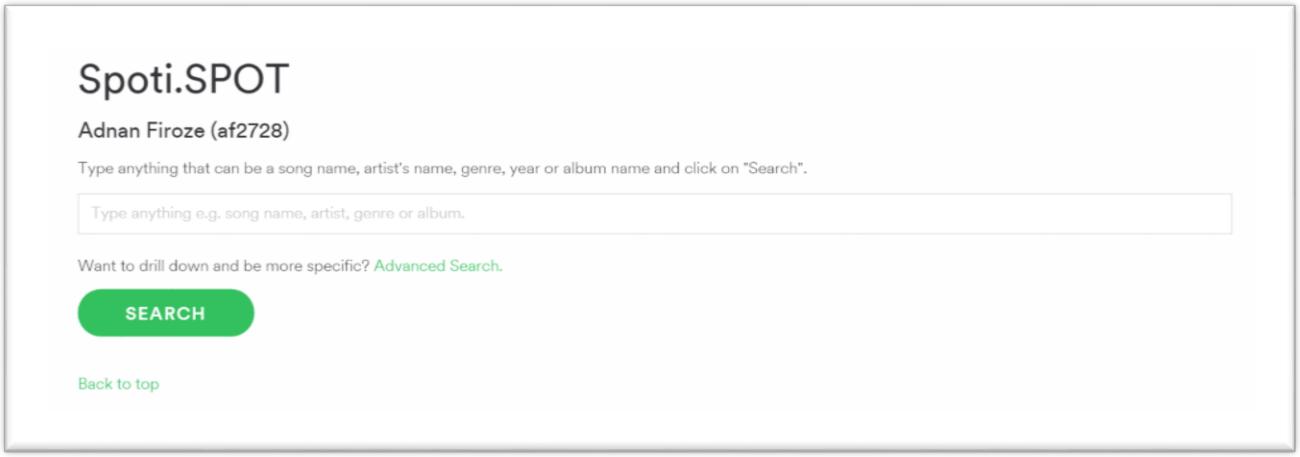
Figure 2: Same search 'All along the watchtower' performed by 'Dave Matthews Band' between 2008-2015 gives us just 2 tracks (great filter!)

Note: You do not need to change any path because all paths that are used are "relative". So, simply open "index.html".

(ii) Heuristics decisions/consideration and analysis:

In this section I will refer to **JAKOB NIELSEN**'s 10 Usability Heuristics for User Interface Design since they are extremely insightful and I tried my best to make the app meet most of the heuristics.

I started with the eight heuristic in consideration titled "**Aesthetic and minimalist design**". Therefore, my initial page is as minimalist as possible (like that of Google's landing page). You only see what is relevant and the extra features are all under the dashboard but the user will automatically see what to do. Screenshot below (this is just the landing page – other options are in "Advanced Search" :


Spoti.SPOT

Adnan Firoze (af2728)

Type anything that can be a song name, artist's name, genre, year or album name and click on "Search".

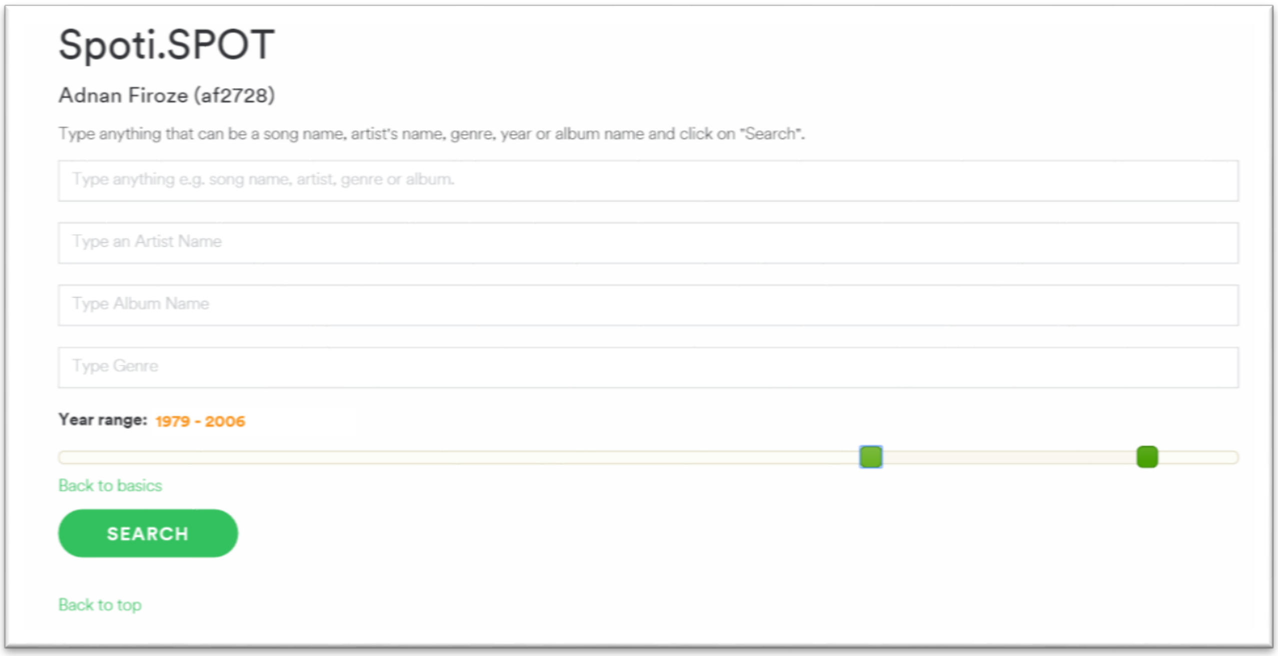
Type anything e.g. song name, artist, genre or album.

Want to drill down and be more specific? Advanced Search.

Figure 3: Spoti.SPOT landing page

As for aesthetics, you can see that I have used the same button format as that of Spotify to be coherent and familiar to the user. However, for my own preference I chose a white design instead of black (used by Spotify). I believe my design meets Nielsen's aesthetics and minimalist design.

Now for the third heuristic i.e. **User control and freedom**, I have made emergency exit in all situations. Samples are shown below. For example, suppose a user wants "Advanced Search" and wants to go back to the basics, he/she can do that readily. See screenshots below:


Spoti.SPOT

Adnan Firoze (af2728)

Type anything that can be a song name, artist's name, genre, year or album name and click on "Search".

Type anything e.g. song name, artist, genre or album.

Type an Artist Name

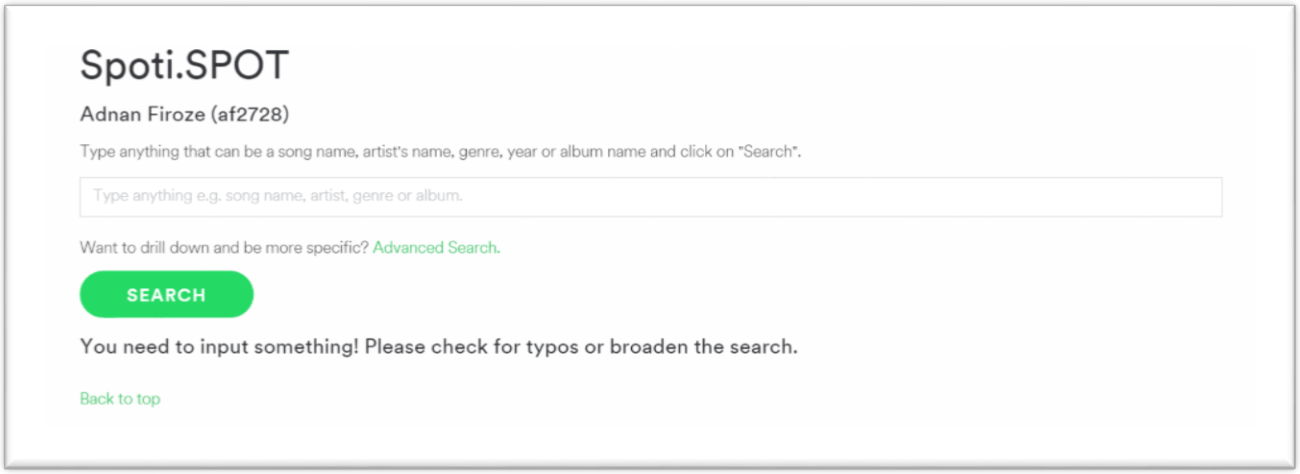
Type Album Name

Type Genre

Year range: 1979 - 2006

Figure 4: Once "Advanced Search" is clicked, you get filters as shown. Also, notice the "Back to basics" link right above the Search button. That will revert back to the basic search.

Another heuristic is to make sure proper prompts are shown to the user and make recommendation. Say a user inputs nothing, then he/she will get a message along with the app so that no page is reloaded. Screenshot below.


Spoti.SPOT

Adnan Firoze (af2728)

Type anything that can be a song name, artist's name, genre, year or album name and click on "Search".

Type anything e.g. song name, artist, genre or album.

Want to drill down and be more specific? Advanced Search.

SEARCH

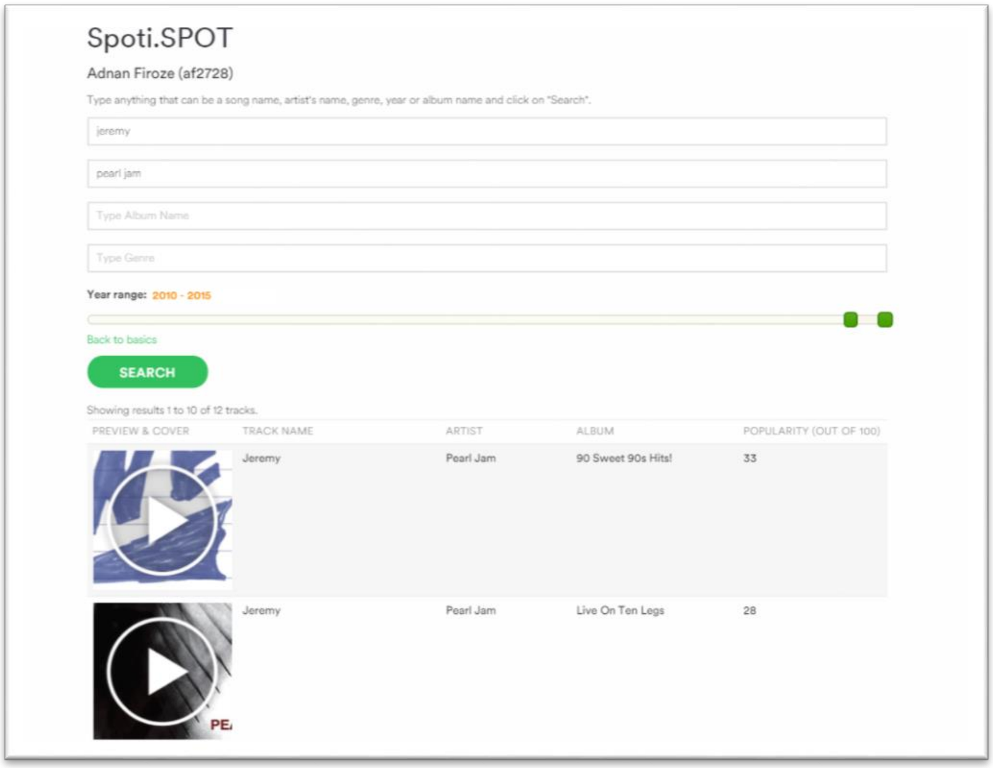
You need to input something! Please check for typos or broaden the search.

Back to top

Figure 5: Non-invasive prompt showing nothing was inputted (saves API call and also the form is present)

Moving on to the 5th heuristic – "**Error Prevention**", I have also considered it. Let's say a query returned 25 results and each page includes 10 records. In that case, while the user is on the first page, he/she will not see the "Previous" button and if he/she is on the last page, he/she will not see the "Next" button.

NOTE: MY PAGINATION FEATURE IS HAND CRAFTED IN JAVASCRIPT+JQUERY WITHOUT ANY PLUGIN – SO A LOT OF WORK WENT INTO THIS PART. Screenshots below:



Spoti.SPOT

Adnan Firoze (af2728)

Type anything that can be a song name, artist's name, genre, year or album name and click on "Search".

jeremy

pearl jam

Type Album Name

Type Genre

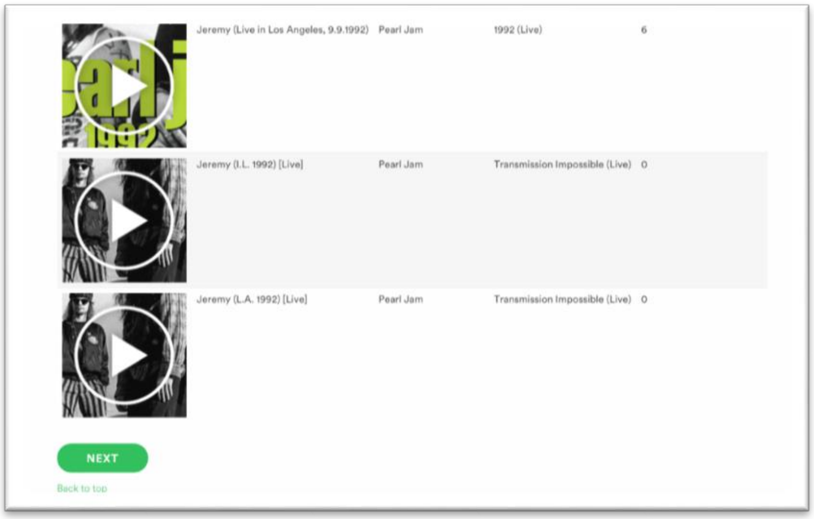
Year range: 2010 - 2015

Back to basics

Showing results 1 to 10 of 12 tracks.

PREVIEW & COVER	TRACK NAME	ARTIST	ALBUM	POPULARITY (OUT OF 100)
	Jeremy	Pearl Jam	90 Sweet 90s Hits!	33
	Jeremy	Pearl Jam	Live On Ten Legs	28

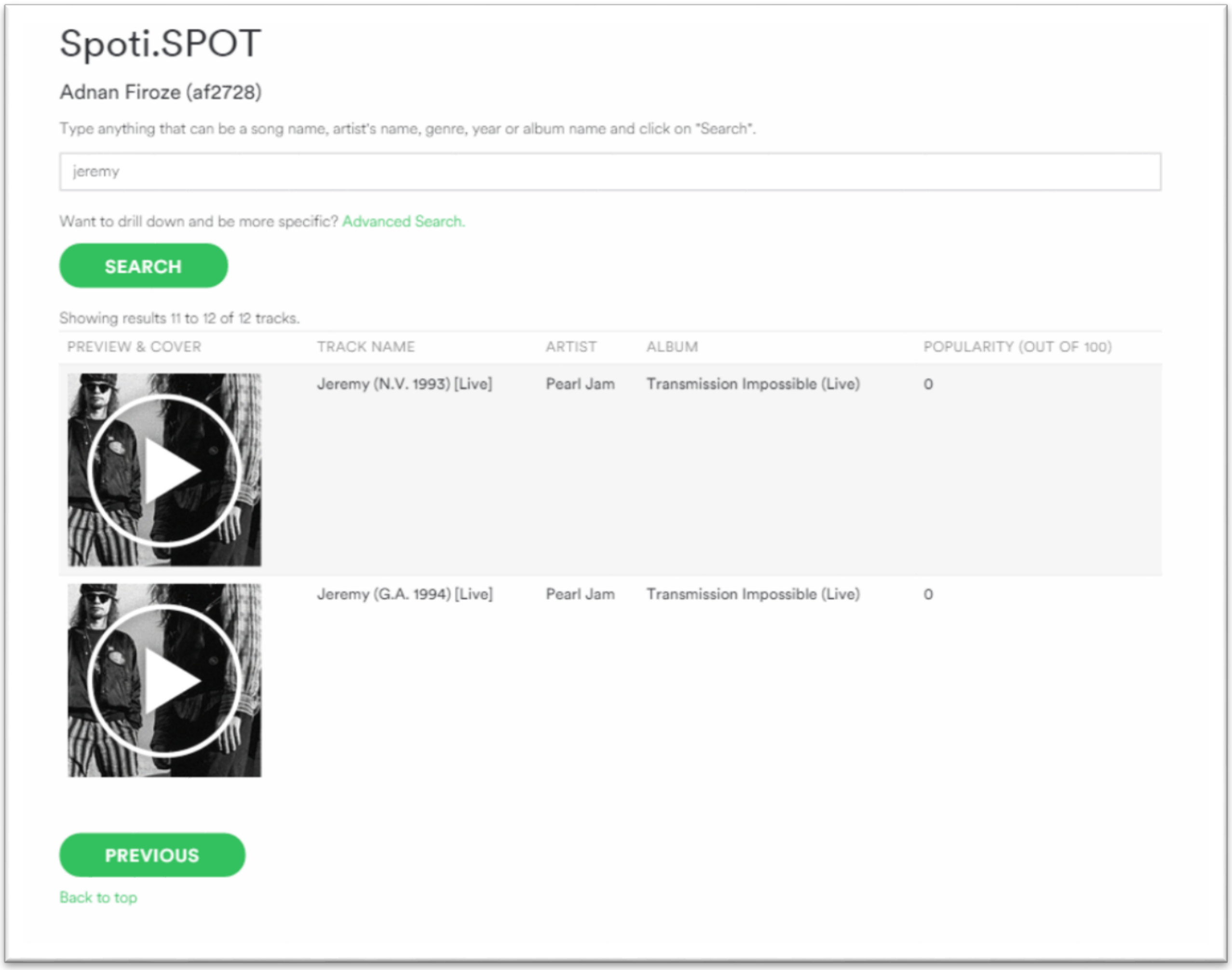
Figure 5: A search of "Jeremy" by "Pearl Jam" during the 2010-2015 range. This is the first page and it returned 12 tracks (which is also mentioned – look closely). As this is the first page, it shouldn't have a "Previous" button. Let's see what is at the end of the page. THE PLAY BUTTONS ARE OVERLAYS ON ALBUM COVERS – THIS DECISION I BASED ON ARTISTS' MARKETING STRATEGY.



	Jeremy (Live in Los Angeles, 9.9.1992)	Pearl Jam	1992 (Live)	6
	Jeremy (I.L. 1992) [Live]	Pearl Jam	Transmission Impossible (Live)	0
	Jeremy (L.A. 1992) [Live]	Pearl Jam	Transmission Impossible (Live)	0

Back to top

Figure 6: The first page has no "Previous" button. Now let's look at a last page.


Spoti.SPOT

Adnan Firoze (af2728)

Type anything that can be a song name, artist's name, genre, year or album name and click on "Search".

jeremy

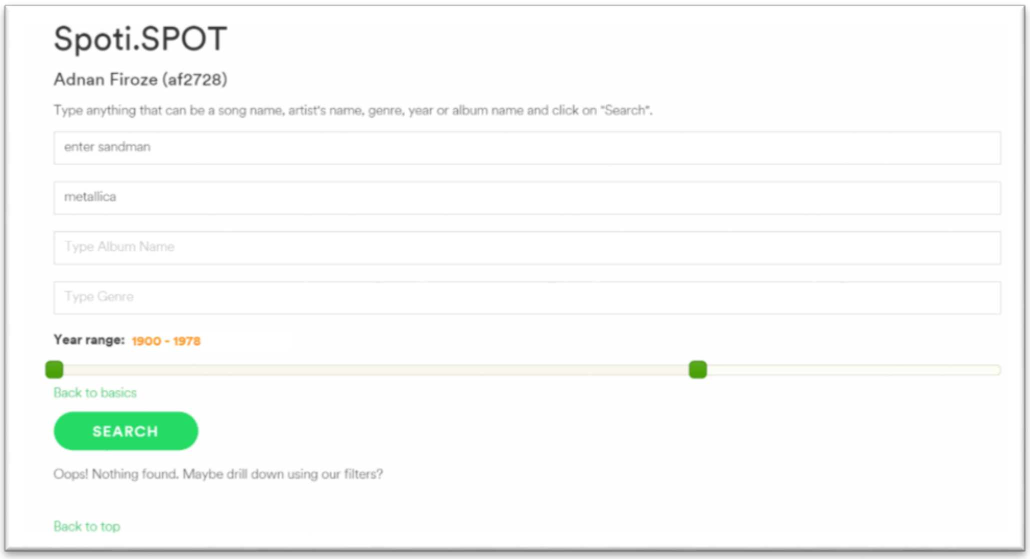
Want to drill down and be more specific? Advanced Search.

Showing results 11 to 12 of 12 tracks.

PREVIEW & COVER	TRACK NAME	ARTIST	ALBUM	POPULARITY (OUT OF 100)
	Jeremy (N.V. 1993) [Live]	Pearl Jam	Transmission Impossible (Live)	0
	Jeremy (G.A. 1994) [Live]	Pearl Jam	Transmission Impossible (Live)	0

Figure 7: Even though each page should have 10 songs, as it is the last page (2nd page in 12 songs returned), we see two songs only and also no "Next" button. This was totally hand crafted without any plugin.

Another error prevention measure is as follows: Say we search for Metal lica's "Enter Sandman" but released in the range of years between 1900 and 1978. We all know (I hope) that Metallica was formed in the 80's. So the API should give us nothing back. Here's what I do to handle it (screenshot below):



Spoti.SPOT

Adnan Firoze (af2728)

Type anything that can be a song name, artist's name, genre, year or album name and click on "Search".

enter sandman

metallica

Type Album Name

Type Genre

Year range: 1900 - 1978

Back to basics

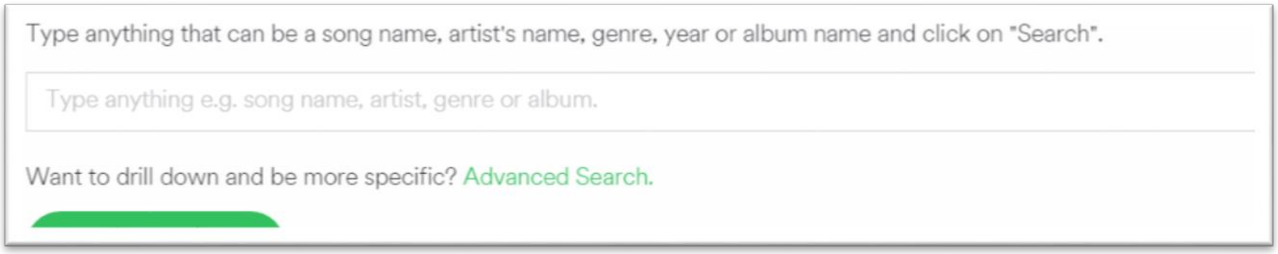
SEARCH

Oops! Nothing found. Maybe drill down using our filters?

Back to top

Figure 8: Situation where query returns nothing (see below Search button)

Next we move to Nielsen's heuristic titled "**Flexibility and efficiency of use**" that gives us the idea of novice users and accelerators. This is almost perfect in my app. If you enter an artist's name in the text field of the "Basic" version, the user will still find songs by that artist but maybe mixed with some tribute and cover songs. So, whatever is input in "Basic" mode (for novice users), we will get relevant results. Let's see an example:

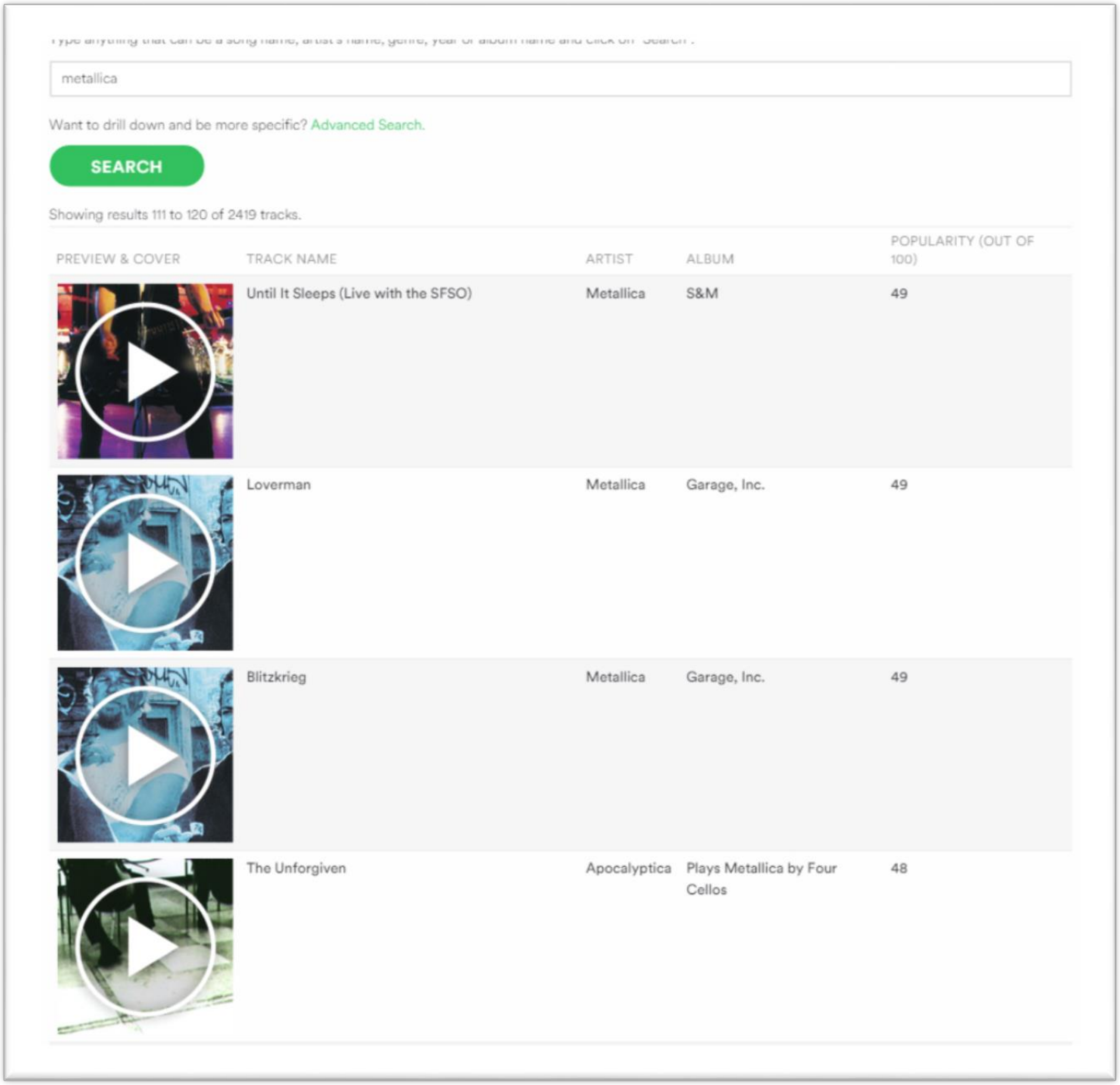


Type anything that can be a song name, artist's name, genre, year or album name and click on "Search".

Type anything e.g. song name, artist, genre or album.

Want to drill down and be more specific? Advanced Search.

Figure 9: Notice what is pre-written in the text field. It's not just track name. Now let's input an artist name there. See next figure:



metallica

Want to drill down and be more specific? Advanced Search.

Showing results 111 to 120 of 2419 tracks.

PREVIEW & COVER	TRACK NAME	ARTIST	ALBUM	POPULARITY (OUT OF 100)
	Until It Sleeps (Live with the SFSO)	Metallica	S&M	49
	Loverman	Metallica	Garage, Inc.	49
	Blitzkrieg	Metallica	Garage, Inc.	49
	The Unforgiven	Apocalyptica	Plays Metallica by Four Cellos	48

Figure 10: When an artist name (Metallica) was inputted in the Basic mode instead of a track name, we found 2419 results and 99% of them are by Metallica. The first track was found as the 114th record that was a cover song originally by Metallica. So, we have met this heuristic.

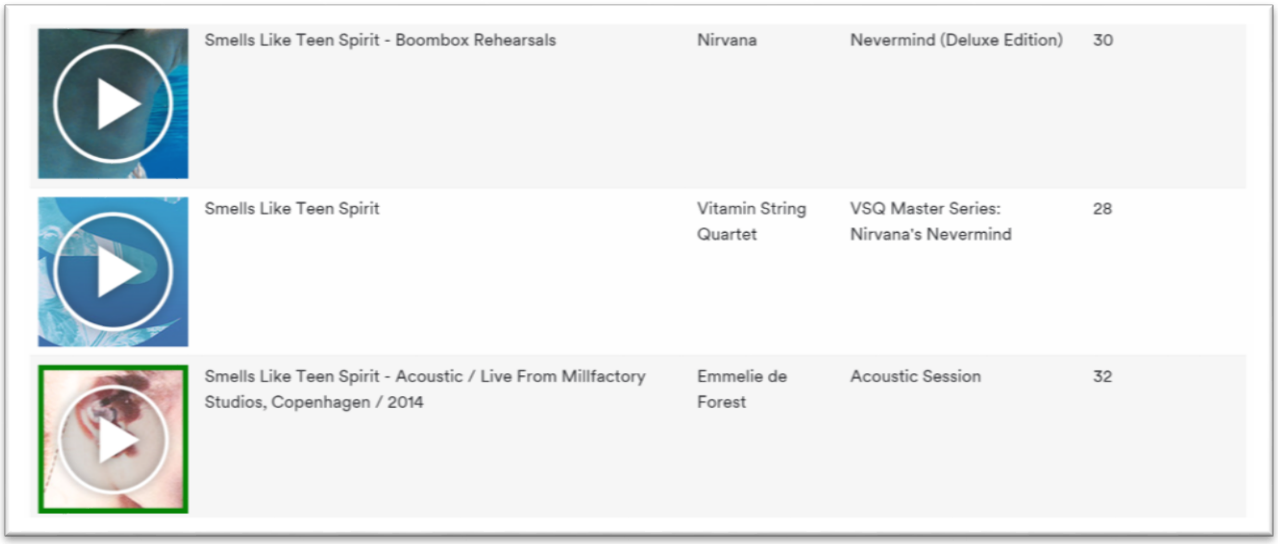
Another example of an 'accelerator' is the simple "Back to top" link at the end of pages of lists.

The previous example is also an excellent example of "Recognition rather than recall" heuristic. Why? Because the user does not need to remember that, "I need to click the 'Advanced Search' link and when the additional fields come up, I need to type 'Metallica' or 'Nirvana' in the 'Artist' field. He can just input 'Nirvana' or 'Metallica' in the single field of 'Basic' search because he still gets everything from whomever he typed. This would be true for 'genres' as well.

The last heuristic by Nielsen is “**Help and Documentation**”. As the app is a small one, there is no separate ‘help’ tab or wiki because that would be unnecessary and as all information is presented to the user on screen – having a separate Help Wiki would violate the “Minimalist” design. Therefore, the design has been taken into account and met with by combining the “**Minimalistic Design**” heuristic.

I have already illustrated in the previous screenshots that the app meets Nielsen’s heuristic of “**Help users recognize, diagnose, and recover from errors**” which suggests using plain language and no code. In fact, my app exposes zero code jargon (not even “record” or “query” etc.). This also corresponds to the heuristic titled “**Match between system and the real world**” because I use words like artist, album, song, popularity etc. that are directly recognizable to anyone interested in music. Since the app is dynamic, it intrinsically meets the “**Consistency and standards**” heuristic since I have not changed any terminology across any use-case.

Now let us move to Nielsen’s heuristic titled “**Visibility of system status**”. In this heuristic, my app does a fairly good job but not exactly perfect. For example, it is obvious for the users to know that the “Play button overlay” will play a song’s sample (see screenshot). When one of those are clicked, we get a green border around that album cover. It could have been implemented better I believe with a different overlay image. Regardless, my design does give the user the visibility of the system’s status:



Smells Like Teen Spirit - Boombox Rehearsals	Nirvana	Nevermind (Deluxe Edition)	30
Smells Like Teen Spirit	Vitamin String Quartet	VSQ Master Series: Nirvana's Nevermind	28
Smells Like Teen Spirit - Acoustic / Live From Millfactory Studios, Copenhagen / 2014	Emmelie de Forest	Acoustic Session	32

Figure 11: System status of song playing using the green border.

Another **acceleration technique** I made is that the user does not have to pause or stop the track that is playing whereas he/she can directly click on another song and it will change to that song.

One extra good heuristic I believe I was cautious about was to put **ALT-text** for all images that include the buttons and also dynamically the myriad of album covers that are returned from queries (based on their names). For an “Accessibility” heuristic, I feel that it was important.

(v) Best Practices:

1. My design is clean and simplistic
2. Not at all image heavy

3. All the Javascript/JQuery and CSS styling are in separate files (has no inline HTML styles)
4. Every image – buttons and even the album covers have “alt” text for accessibility.