**Ladbrokes Coral NodeJS test**

**Introduction**

This exercise will use the SoundCloud API to fetch data, to store it and return to the user.

SoundCloud is a platform that provides users access to a library of tracks. These tracks can be embedded and played on other web pages.

The SoundCloud API allows you to query the tracks.

Further documentation can be found here:

https://developers.soundcloud.com/docs/api/reference#tracks

Do not use any pre-written SoundCloud APIs, this exercise requires that you make a direct http call.

You are allowed to make use of any other library you see fit to help you complete the task.

Required develop solution language: Node.JS.

Required Database: MongoDB or any other NoSQL database

Logger: Winston

When asked for a client\_id please use: pCNN85KHlpoe5K6ZlysWZBEgLJRcftOd

Example:  
http://api.soundcloud.com/tracks/?q=pixes&client\_id=pCNN85KHlpoe5K6ZlysWZBEgLJRcftOd

**The Task**

**A.**

1. Create an API endpoint that receives the following parameters:

* band\_name
* secret\_key

This API will internally make a call to the SoundCloud API and return:

* The 3 most liked songs
* The 3 least liked songs
* In a sorted object (descending)
* Data for each individual song should be
  + title
  + description
  + likes\_count
  + tag\_list

2. Log each request in a NoSQL database.

Each logged database record should contain:

a. Request time in 24h:mm dd/mm/yyyy format (GMT +2).

b. Band name.

c. SoundCloud returned formatted object Or an error message.

3. Errors: In case of an error in band name (band can not be found), in a secret key, or a network error return an appropriate error message.

**B.**

1. Build a script that runs once on predefined array of 5 bands (of your choice).

Reuse the relevant part of section A.1 to get the bands’ 6 tracks.

2. Sort the **bands** from most liked to the least likes songs.

3. Print to those results to t alogger

4. End script.

**C.** Please add a short explanation how to run the solution and please specify any assumptions you have based your solution on, if such assumptions have been made.

**What we’re looking for:**

* Well written code broken into logical components that interact with each other
* Avoid a 1-class implementation.
* Functions brief documentation.

**Good Luck!**