Full Stack Developer Test

The aim of the test is to have a prototype of an interactive sound recording metadata matcher.

A sound recording is the fixation of a musical performance event. It is identified by the International Standard Recording Code (ISRC). Other metadata that represent a sound recording are: title, artist, duration, etc.

We want to match sound recordings from an input report against the sound recordings that we already have in our database. This matching will be interactive. First you should find the candidates and then provide a user interface where an operator could choose the right candidate if any.

Part 1: Getting the candidates (duration 1 or 2 hours)

Firstly, the metadata from `sound_recordings.csv` should be stored in a database. Then, each sound recording in `sound_recordings_input_report.csv` should be matched against that database generating a similarity score. Matching results should be stored in the database, too.

The matcher has to consider all metadata fields provided to match and sort the results. As the final matching decision will be taken by a human, it’s preferred to have false positives to false negatives.

Instructions

- Scripts should be written in Python.
- PostgreSQL or MongoDB can be used as a database.
- Instructions should be provided on how to execute your code.

Questions

1. Describe briefly the matching method chosen.

2. Imagine that your database has 10 million sound recordings, do you think your solution is still valid?

3. If not, what would you do to improve it?

Part 2: Interactive matcher (duration 1 or 2 hours)

To make use of `Part 1`, we need a user interface that, from one hand it will show the metadata from the input report and from the other, the database candidates. Obviously, the candidates will change depending on the sound recording selected from the input report. Candidates should be sorted by the similarity score calculated in `Part 1`.

Once a user selects one of the candidates, the sound recording should disappear from the input report list.

Instructions

- The back-end has to be written in Python. You can use frameworks such as `Flask` or `Django`.
- For the front-end you can use any of these javascript libraries: `ReactJS`, `VueJS` or `AngularJS`.
- The UI has to be in a single page.
- Instructions should be provided on how to execute your code.

Questions

1. Describe why you choose this layout.

2. What would you do to improve the user experience?