Logikcull Take Home Coding Assignment - Core Services Engineer

Summary

Congratulations! Based on your experience and interviews-to-date you've been selected for the next step in the Logikcull recruiting process, the take-home coding assignment! Consider this a chance to show off your coding chops, experiment with a new framework or library (though feel free to stick with what you know too, it might be new to us!), and have a little fun throwing together a small API that meets the requirements set forth below.

Expectations

The assignment is purposefully open-ended. With this position you'd primarily be working with the Rails framework, so homework should be in Ruby using Rails.

On the whole, we recommend spending up to 8 hours working on it (depending on experience level and familiarity with technology).

We value your time! Irrespective of what happens moving forward with your API, you will be compensated with a $200 Amazon Gift Card for finishing this assignment. Treat yourself.
There are no wrong solutions. If you make it to the next stage of the interviewing process we will use what you sent us a starting point for discussing how you like to approach problems, what abstractions you've made, technology choice etc.

Final Product Delivery

Please zip up your project with basic setup instructions to run the API. Email the zip along with any notes to Gazina. Note that you may have to rename the zip file extension to .txt in order for it go through.

Alright, you caught us, we want to give you a chance to show off how great you are at writing clear, communicative documentation, even when the specs aren't 100% spelled out.

Assignment

Help! Logikcull Engineer Greg is a self proclaimed hipster (does this still mean anything in 2020) with a burgeoning record collection numbering in the tens of thousands. His current pen and paper approach to managing the collection isn't scaling well and he needs a solution to help him manage his pride and joy.

Your mission, if you choose to accept it, is to build an API (REST is preferred) for Greg to build a front-end client on top of so that he can manage his records.

Use whatever technology you are comfortable with and feel free to make it as polished or minimalist as you'd like (according to your taste). If you aren't familiar with records, he needs to track/store the following fields per-record (at a minimum): Artist, Album Title, Year, and Record Condition.

When Greg builds a frontend client it must meet a few requirements:

1. The application must be able to fetch and display Greg's albums in a paginated fashion.
2. These fields must be visible for each album: Artist Name, Album Title, Year, and Record Condition
3. Greg’s frontend client must be able to edit and update all the fields on an album, including the artists associated with the album.
4. As Greg’s frontend client, I should be able to update the Artist's name. Meaning if I update an Artist's name on one record, it should be updated across all records associated with that artist.
5. As Greg’s frontend client, I want to filter my albums using a search query. The results should update as I type.
6. As Greg’s frontend client, I want to display the most common word across record titles across the record collection.
7. As Greg’s frontend client, I want to display the number of records released by an artist each year

You must build an API keeping in mind the requirements Greg must meet as well as common requirements for a CRUD UI.

Add any bells and whistles you think would improve user experience. Your main objective should be to create a complete, thoughtful solution that meets the requirements and is a good representation of your coding ability and style.

That’s it! Have fun!

If you have any questions please feel free to Gazina. We are always striving to improve our documentation and clear up anything that might be unclear. So please let us know where we can improve.