Development Activity

Welcome

Congratulations and thank you for your continued interest in Emotive! We'd like to proceed to the next round of our interview process. There are two options from here:

1. Perform this simple take-home activity and return it ahead of your next interview
2. Defer this until your next interview

This decision is completely up to your preferences and it will **only shape** the interview process from here. If you complete this activity and return it, a portion of your next steps will be reviewing your work and pairing on some changes. If you prefer to wait until the next round to complete this, we ask only that you know which direction you'd like to go and we'll pair with you on implementation. The goal of giving you the option is to align with the work setting you are most comfortable with. Either option gives us insight into your working style and how it could potentially be applied to the team at Emotive.

What You Can Expect in Your Interview

Take-Home Activity

Be prepared to:

- Discuss what you chose to build (including your idea, what you focused on, and why)
- Review your code with an Emotive engineer
- Pair with an Emotive engineer to fill out your API
- Bottlenecks, scaling issues with approach taken along with proposed solutions

Real Time Activity Selected

Be prepared to:

- Pair with an Emotive engineer and quickly start building your API.
- Discuss your idea and your process while pairing on initial implementation and subsequent expansion
- Bottlenecks, scaling issues with approach taken along with proposed solutions

The Activity

Build a simple API in Python, using Django or Flask. This API should integrate with a third party service or a database and accomplish something interesting, but also shouldn't take terribly long to implement - endpoints for one or two resources would be sufficient. This activity shouldn't take more than an hour or so. If this doesn't seem like much time - you're right! We expect that you'll have to focus on one of a few areas for your API:

- The API design and interface (REST vs. GraphQL)
- Integration with a third-party API OR Integration with a database

Requirements

1. A clear problem statement - what do you intend the API to accomplish?
2. Dependencies for running the API
3. A code repository link from which your interview can checkout the code **OR** a .zip file containing the source code
4. A README file in the code root containing instructions for running the API

Some Ideas…

Interesting APIs

- NASA Open APIs
- OpenWeather API
- Polygon.io Stocks API

Plus anything more you can discover

Databases

- SQLite
- PostgreSQL
- MySQL
- MariaDB

PostgreSQL is provided in our skeleton projects, but another database is fine if you'd prefer.

Projects

- A stock recommendation service
- A green/red alert for outdoor activity safety
- A Mars Rover camera viewer