Chattermill Full Stack Coding Challenge

Instead of building something dull and unrelated to what we do here in Chattermill let's create a simplified version of Chattermill product!

So... let's imagine we are at the very beginning of Chattermill history. Recently we conducted several customer development interviews asking our potential clients how they analyze their customer feedback (let's say their product reviews) and what kind of tool could be helpful for this. We realized that we need to build a very simple product that would allow them to track experience of their customers by giving them a dashboard that displays metrics based on their customer feedback.

To build an MVP we gathered a dataset based on public data which represents Google Play and Apple App Store reviews for various mobile apps popular in the UK. We applied our machine learning algorithms to this data to extract themes (which are various aspects of our client's business like payments, delivery, product quality etc.) of each review and sentiments of those themes.

For our MVP we need to build a frontend with a single screen that would display couple charts and a backend that would serve API for our frontend.

We'll discuss the exact functionality of our MVP a bit later and now let's consider our data model.

Data model

Our data model comprises 3 entities: reviews, categories and themes.

Review - is a review of a product/service e.g. app review in the Apple App Store or Google Play. Review has the following fields:

- id - unique identifier of a review
- created_at - date and time a review was created at in ISO 8601 format
- comment - text of the review
- themes - set of review themes identified by our machine learning algorithms. Review theme has the following fields:
    - theme_id - id of the theme
    - sentiment - emotional characteristic of a review's theme which can be negative (-1), neutral (0) or positive (+1)

Category - is some broad topic related to the business of our client (it can be a product quality, delivery, payments etc.). Category has 2 fields:

- id - category identifier

- name - human-readable category name

Theme - is in fact a sub-category (e.g. in case of payments it can be topics related to a particular payment method). Category contains several themes. Theme has the following fields:

- id - theme identifier
- name - human-readable theme name
- category_id - id of the category this theme belongs to

You received *dataset.zip* which contains JSON representation of our dataset. Please use these files to populate data storage of your choice.

Now let's get to the actual features we're going to build for our MVP!

Dashboard screen

Dashboard screen is the only screen of our frontend and it contains filters and charts. Here are the filters we need to implement:

- Filter by theme
- Filter by category
- Filter by phrase in a product review

In theme and category filters user can select one of the human-readable options from dropdown. Phrase filter is a text input where we can type an arbitrary substring of a comment field of a product review and thus filter reviews containing this substring. When we add or remove a filter our charts are getting updated.

We need to display 2 charts. One of them displays average sentiment with breakdown by category and another by theme. We recommend to use Highcharts JS to display these charts.

Backend requirements

Besides API that your frontend needs you will need to implement one additional API endpoint that will be used by our clients to provide new reviews. This method creates a new review in your database and its request will contain a comment and themes with their sentiments. We will use this endpoint to populate your database with more reviews and test how your API performs for bigger datasets than the one we provided.

Feel free to choose any API protocol that feels right to you for this task as well as any database.

Keep in mind that one of the highest priorities for us is responsiveness of our charts because this is an analytical tool that will be used by our clients and they will be playing with different combinations of filters to find insights they need and they expect to see results right after updating the filter.

Another obvious requirement is a durability of data - our clients rely on Chattermill in understanding of how their customers feel and we think every single customer should be heard. For us it means we need to reduce the probability of losing any events in our system to the bare minimum.

Non-functional requirements

This coding challenge is an opportunity to show off your development skills and ethics. Please provide proper tests, readme (on how to build, run and test your code and maybe your design considerations) and comments (when required) and use all the tools you need to ensure quality up to your standards. You can optionally deploy your solution to a cloud platform of your choice. If you choose to deploy it to kubernetes please provide us your k8s specs.