Chattermill Frontend 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 we need to build a very simple product that would allow them to see what their customers are speaking about in their feedback and what emotions they express. We decided to extract different aspects of our client's business mentioned in a product review and simply display them besides each product review as well as emotions that their customers express when speaking about those aspects.

We decided to build an MVP with 2 screens:

- Login screen - here our clients simply sign in with their username/password.
- Feed screen - on this screen we show feed of customer feedback with themes and emotional sentiment displayed besides each product review.

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

Data model and API

Our data model comprises 2 entities: reviews and themes.

Review - is a review of a product/service e.g. app review in the Apple AppStore 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)

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

- id - theme identifier
- name - human-readable theme name

We have a REST API for these entities. Swagger documentation for the API can be found here https://frontend-task.production.cloud.chattermill.xyz/swagger/index.html.

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

Login screen

Login screen is a separate route in our app that contains a form with username and password fields and a login button.

Our server implements JWT authentication and one needs to use **/login** method in order to obtain a JWT token to call our API. Please use the following username and password to authenticate with the server:

Username: `chattermill`
Password: `SuperSecretChattermillPassword!`

Documentation for Login API can be found in our Swagger as well (and you call /login method from the Swagger of course).

Once you obtain a token you can use it to play around with our API right from the Swagger by pressing the Authorize button and specifying token in the following format: ***Bearer %token obtained from /login method %***.

Please note that the Login API response contains an expiration field and our tokens live only for 1 hour thus you need to handle expiration of a token properly.

Feed screen

Feed screen consists of the filter panel and the feed itself.

In the filters panel we can add and remove filters. We need to implement a filter by theme.
In a filter user can select one of the human-readable options from dropdown.
When a filter is updated the feed updates accordingly.

The feed is simply a list of components displaying product reviews. Each component in a list displays a comment field of a product review as well as themes with their sentiment (tip: very concise way to visualize sentiment is by using emojis).

As you can see from our Swagger docs our API supports pagination so we need to take care of pagination on a frontend as well (bonus points for infinite scroll).

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 use any libraries/components out there to implement the needed functionality.