The London Feed - Developer Test

Specification

The task given is to implement a basic front-end react application to display information gathered from a simple REST API and socket endpoints, as well as allowing the user to submit an item as a favourite. Favourited items should be then available to view. A basic mockup has been provided for general layout of the application.

Install + Run

The application is available to view here https://london-feed.herokuapp.com/
The source is available on GitHub : https://github.com/nileahtobhair/london-feed

Prerequisites create-react-app + yarn .

Run : % yarn install. % yarn start.

Implementation Decisions

As the information required to be displayed was not specified (beyond the mockup) I kept it condensed, with links to data source for reviews and gifs. The styling in extremely basic but clear.

I elected to handle data received from the socket streams differently for different data types. For travel updates I appended them to the view immediately on receipt of the information. I felt this was a natural way to view this type of information. I did the same for the cats gifs. For the reviews I implemented a user click to see the newly received updates. I did not handle duplicate items.

There is a couple of things that I felt were beyond the scope of what I felt was a necessary for this test but would recommend to implement going forward to production.

Additional Suggested Functionality

Sort by for reviews

Enough information is given in the review data object to allow for multiple sort by (price , location , type of food , etc) which could improve user experience. I felt it was a little beyond the scope of this project but could be implemented on the front-end with JS array sort.

Expand on API

Additional functionality for the api would greatly improve the functionality of the project. Getting a single item by id , storing star information, allowing for unstaring etc. Also suggested would be to use the twitter api directly for TFL tweets . Basic tweet displays can be done without 0auth or much implementation. Loading new tweets in real time is handled by the API.

Captcha/Authentication

Currently nothing prevents the user from continuously submitted a POST from for the star functionality. Going forward it would be suggested to only allow signed in users to access the feature or implement a Captcha to confirm user is an authentic user.

Pagination

Views will be appended continuously with information gathered from the socket endpoints. Pagination or a max number of displayed items would be suggested going forward.

Testing

I elected not to include testing as this project will not be used and the sample server also did not include testing.