Technical Assessment

Exercise 1 - Fetching data from an API

Write a React hook called `useAPI` which takes one argument `url: string` and returns an object with the properties: `data`, `errorMessage` and `loading`. This function should handle fetching data from the given `url` and returning this as a JSON formatted object to the user. The `loading` property should be `true` whilst the API call is in progress and `false` otherwise. If there are any errors, these should be reported to the user. Writing types for the hook is advised.

Exercise 2 - Using your hook and displaying data

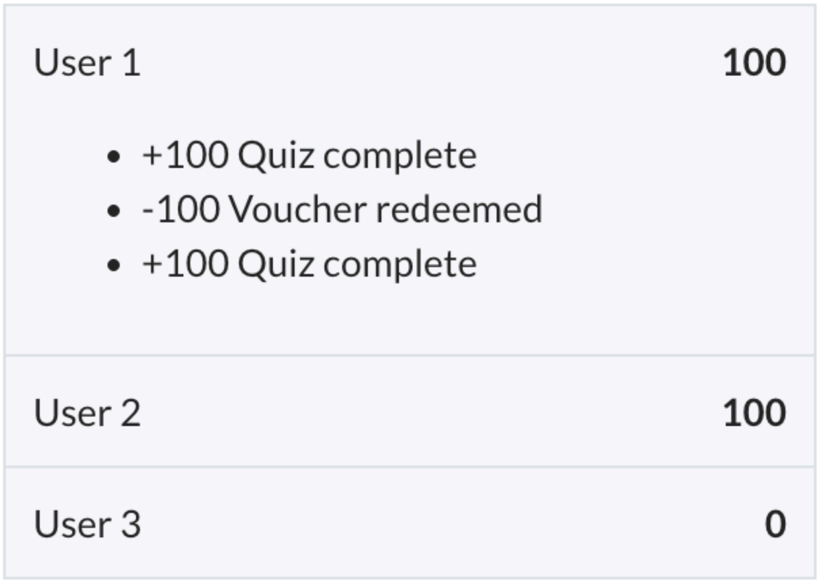
In your `App` component, use the `useAPI` hook you just wrote to hit the URL `http://localhost:3000/electrons`. This should return an array of `ElectronStreamEvent`s, (Electrons are the name of our reward currency) each of which represents a single instance of a user earning or spending their Electrons (this event will contain a `userId`, `timestamp`, `reason` and `pointsAdjustment`). For each user, **sum up** the total number of Electrons they have, and do a `console.log` of the results. Any errors, or loading states from the `useAPI` hook should be surfaced to the user.

Exercise 3 - Building a custom component

Instead of using `console.log` to display our user's Electron balance we want to display the UserId and their total Electron balance on the page. This information should be rendered as an accordion which has the following behaviours:

- Clicking on any of the elements of the accordion will expand that element to display the **full electron stream** for that specific user (no need to worry about animations)
- When any element is expanded, all other expanded elements should be collapsed (i.e, only one item can be expanded at any time)
- Clicking on an expanded element will close it.

Designs for the accordion have been included below and should be adhered to as much as possible.



Property	Value
Background colour	#F8F8FA

Border colour	#F8F8FA
Padding	10px
Max width	300px

Exercise 4 - Extending the functionality of your application

Extend the application to include the ability to add new events to the electron event stream. These new electron events do not need to be persisted across sessions or page refreshes, and they do no need to be sent to an API. The functionality should allow the inputting of a given UserId, as well an electron amount, with some way of determining if this should add to, or subtract from, the total electron balance. When a new event is added, the UI should update such that all the information from the API as well as any locally added events are visible.