Frontend Core

Problem version 2.2.1

Don't spend more than 1 hour on this problem. You are judged on the correctness, robustness and quality of your code. This problem uses ReactJS and Typescript.

There are 5 components below. Each component gets data by sending a POST request to `METRIC_ENDPOINT` with a 'query' generated by `genQuery()` function in the request body. You can assume that each component is completely destroyed and recreated when its `timeRange` property changes.

Fetch data for each component:

- The first time the component is rendered
- After each `refreshInterval_Secs`

Show `Loading` when component is loading data for the first time only.

If I set `NODE_ENV` environment variable to `"production"` in Webpack during compilation time, then `PROD_SVR` url is used, otherwise `DEV_SVR` url is used to fetch data.

```
import * as React from 'react';

const DEV_SVR = "https://dev.dummy-svr.com";
const PROD_SVR = "https://prod.dummy-svr.com";

const METRIC_ENDPOINT = "/metrics";

function genQuery(timeRange: string, componentName: string, seed: number) {
    return `SELECT ${timeRange} WHERE c = ${componentName} AND x = ${(seed%7)==0?'true':'false'}`;
}

function Loading() {
    return <h2>Loading</h2>;
}

interface IProps {
    timeRange: string;
}
```

```
function C1(props: IProps) {

    const refreshInterval_Secs = 60;
    const query =  genQuery(props.timeRange, "c1", Math.random());

    const data = //TODO fetch data;

    return <>Hi {data}</>;
}


function C2(props: IProps) {

    const refreshInterval_Secs = 10;
    const query =  genQuery(props.timeRange, "c2", Math.random());

    const data = //TODO fetch data;


    return <>Hello there {data}</>;
}


function C3(props: IProps) {

    const refreshInterval_Secs = 15;
    const query =  genQuery(props.timeRange, "c3", Math.random());

    const data = //TODO fetch data;

    return <>Charlie {data} Tango</>;
}

function C4(props: IProps) {

    const refreshInterval_Secs = 42;
    const query =  genQuery(props.timeRange, "c4", Math.random());

    const data = //TODO fetch data;

    return <>A fox jumped {data}</>;
}

function C5(props: IProps) {

    const refreshInterval_Secs = 30;
    const query =  genQuery(props.timeRange, "c5", Math.random());

    const data = //TODO fetch data;

    return <>{data} is king</>;
}
```