Assignment and Instructions:

A retailer offers a rewards program to its customers, awarding points based on each recorded purchase.

A customer receives 2 points for every dollar spent over $100 in each transaction, plus 1 point for every dollar spent between $50 and $100 in each transaction.
(e.g. a $120 purchase = 2x$20 + 1x$50 = 90 points).

Given a record of every transaction during a three month period, calculate the reward points earned for each customer per month and total.

- Use React JS (do not use TypeScript)
- Simulate an asynchronous API call to fetch data
- Make up a data set to best demonstrate your solution
- Check solution into GitHub

Checklist

a. **WE ARE LOOKING FOR ROCKSTAR, USE THIS ASSIGNMENT TO SHOW US THAT YOU ARE SENIOR DEVELOPER**

b. **Goal of the assignment**: showcase great craftsmanship in area of REACT JS. Solution should be designed in a way that shows healthy architecture (solutions with 2-3 files will not be accepted). Solution should show the data flow and good practices in working with backend from the UI perspective

c. **Time required:** anywhere between 3-6h depending on proficiency

d. **Technical guidelines:**

   i. Can use https://create-react-app.dev/ to get started quickly
   ii. avoid additional frameworks and Typescript – only REACT JS
   iii. Make up a data set to best demonstrate your solution (can use tools to generate data or manually create some data)
   iv. Check solution into GitHub and share the URL(Make it Public)
   v. Plan your code composition structure (e.g separate service to simulate API request, separate file to calculate points, split UI into few readable files, etc.)
   vi. No hardcoded / magic numbers. Use constant definitions/dynamic props
   vii. Code should be clean, nicely formatted, easy to read and understand and follow best practices
   viii. UI needs to be reasonably OK. Not expecting a fancy UI.
   ix. No errors or warnings in runtime
   x. Code must compile and run, there should be an instruction in README.md file how to run the application.
   xi. Include unit tests
   xii. Handle errors after loading application
   xiii. Handle data loading – e.g. spinner