**Please review this document in its entirety prior to starting the initial stage of the interview/assessment process.**

**Coding options based on candidate skillset. Please update your recruiter as to what you completed it in.**

1. Candidates need to wrap the assessment in Spring Boot if they are Web API resources
2. Candidates need to use ReactJS if they are UI resources
3. If Javascript, try and create using ReactJS. The end client will take in account the lack of experience but is looking to see if the resource can adapt to a new technology and follow best practices and standards.
4. Do not use Typescript

**Top reasons applicants fail the initial stage of interview process/assessment.**

1. Unit Tests-these are required
2. Keep the implementation as simple as possible so it’s easy for someone new to read and understand. Cleanly extract and separate out common re-usable logic into separate files wherever possible.
3. Follow component-based software engineering principles when completing the assessment.
4. Do not copy/past code. The client has reviewed enough assessment over the years to quickly identify this.
5. The creation of clean code is mandatory, please review the following article for clarification:

[**https://cvuorinen.net/2014/04/what-is-clean-code-and-why-should-you-care/**](https://cvuorinen.net/2014/04/what-is-clean-code-and-why-should-you-care/)

1. When feasible, utilize the newest technology and features. Example: Javascript features like ES6 built in methods. Java version 8 or higher vs version 6.
2. Variable Naming Convention. Use appropriate names for the variables used. This increases readability for other developers
3. Avoid using unnecessary 3rd party libraries.

**Things to evaluate and consider prior to taking the assessment if you choose to do so in React.**

1. For react apps, please structure and organize the project properly with separate folders for **components,** **CSS Styling** etc. and store all the relevant components for the application in the Components folder and styles in the styles folder etc. not outside in the main app.
2. Do not put complex Javascript logic inside HTML tags unless its basic JS logic, if the logic is complex, it should be separated out into its own file or component if the component needs to be further split into child components.
3. Keep the implementation as simple as possible so it’s easy for someone new to read and understand. Cleanly extract and separate out common re-usable logic into separate files wherever possible.
4. **DO NOT forget unit tests-they should be in a JS folder-titled test**
5. If needed separate out the CSS files whenever possible to be more relevant to the component it applies to
6. **https://www.codeinwp.com/blog/react-best-practices/**

**Pluses**

7. Try to use React hooks and custom hooks wherever possible to promote code re-use.

**Things to evaluate and consider prior to taking the assessment if you choose to do so in React.**

* 1. **https://www.e4developer.com/2018/08/06/spring-boot-best-practices/**

**Assessment-Two options for UI and Web Api Developers**

**UI Developer**

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 over $50 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

**WebAPI Developer**

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

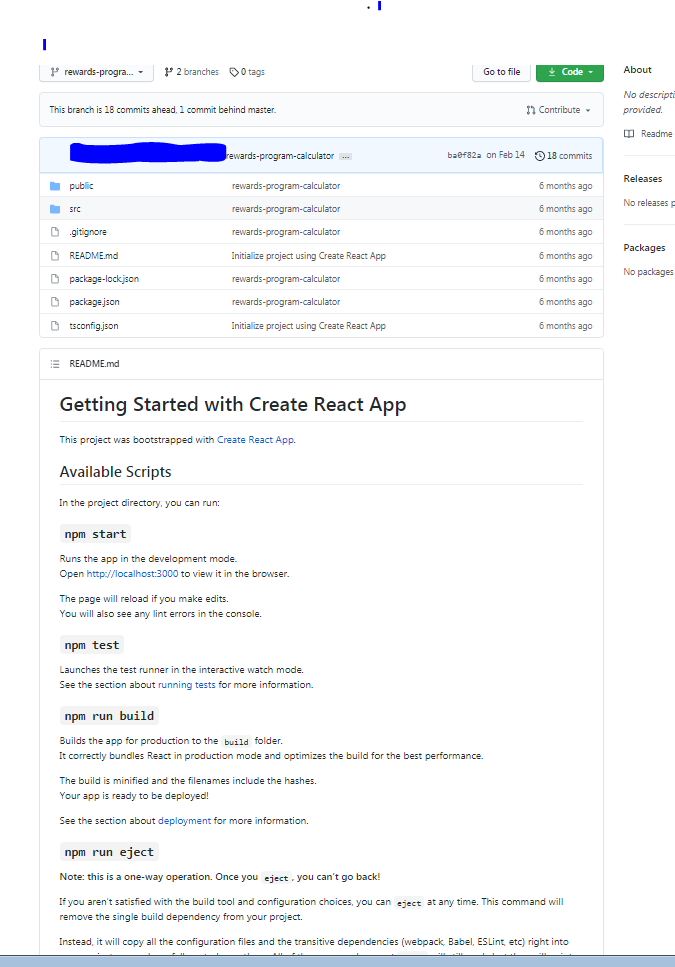
A customer receives 2 points for every dollar spent over $100 in each transaction, plus 1 point for every dollar spent over $50 in each transaction

(e.g. a $120 purchase = 2x$20 + 1x$50 = 90 points).

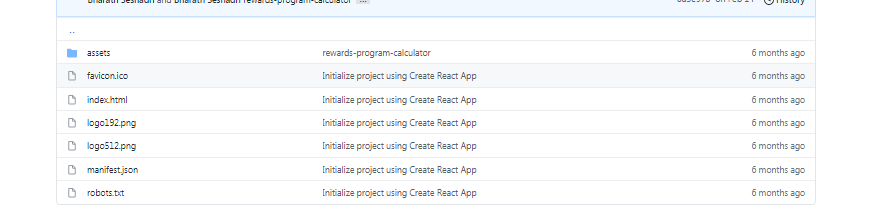
Given a record of every transaction during a three-month period, calculate the reward points earned for each customer per month and total.

* Solve using Spring Boot
* Create a RESTful endpoint
* Make up a data set to best demonstrate your solution
* Check solution into GitHub

**React Final Product and set up.**



**Public Folder within react.**



**SRC Folder within React**

