GSynergy Typescript React Challenge – Progressive Web App

Introduction

Thank you for your interest in advancing your career at GSynergy. We look forward to a productive and insightful interview process that informs both you and us if we are going to be a good fit for building great software together.

All our applications are developed using React. We are a fast paced, high productivity team, where all our developers have high level of autonomy and ownership. We therefore require that all our teammates have strong development skills, and are fairly comfortable using React. Since everyone works from home, we require that everyone have strong oral, written, and visual communication skills.

The following challenge is meant to assess your

1. foundational abilities in using React
2. coding and development style
3. ability to structure code for reusability, maintainability and testability
4. ability to communicate your work effectively and efficiently

If you have the appropriate level of proficiency, the challenge should take you between 2 and 6 hours of effort, but if you get carried away, we will not penalize you for your passion :)

General Rules and Guidance

1. Do not seek assistance from anyone else, or plagiarize solutions from the web.
2. Structure and write your code as if you are writing it for production.
3. We highly recommend you employ test-driven development, and demonstrate your use of TDD through code commits.
4. Ensure your code is working before submitting it for review. You will not get a second chance.
5. You must use Typescript.
6. You must use a state management library such as Redux, etc.
7. Please reach out to us at careers@gsynergy.com if something requires clarification. We may take up to 24 hours to respond.

How to deliver

1. Create a public Github repository named as: *GS######_YourFirstName_YourLastName*, where ###### is a 6 digit random number of your choice. The repo should have your entire code, and any assets.

We recommend you start with creating an empty repo, and commit regularly, as you would when writing production grade code.

2. Create a brief readme.md file with the following content:
    a. Instructions about how to run and test your code.
    b. List elements from the challenge that you think you have done well, and that exemplify your proficiency. Please describe why you chose those elements, and how they demonstrate your proficiency.
    c. List what you would do to improve your solution if you had 4 more hours available for this task. Describe why you would do those things.
    d. Optional: Any feedback about how we may improve this challenge.
3. Deploy the app to Google Firebase, AWS Amplify, Cloudflare, Netlify, Vercel, or any similar service. The primary requirement is that we should be able to run your deployed code without requiring any additional setup.
4. Create a 2–5 min screen recording demonstrating your work and share it on a video or file sharing site which does not require us to download the video file for viewing it. You must use your own voice to describe your work. You may use software of your choice to create the video. If you don't know what to use, we recommend using Zoom meetings to record your own screen; it is easy and the video files are small.
5. When done, share the links to the repo, the hosted app, and your demo video with careers@gsynergy.com. Test your links before you send out your email. Depending on the volume of submissions, we may not follow up with you if the links don't work.

ReactJS Challenge: Data Viewer

You will be developing a progressive web app for manipulating and analyzing data created in your app. The app will include two pages to add dimension members, one page with an AG-Grid to enter measure data, calculate some expressions, and then use conditional formatting to highlight outliers, and one page to show a chart.

Refer to the following for libraries you may use

1. AG-Grid: https://www.ag-grid.com/
2. AG-Charts: https://www.ag-grid.com/charts/
3. Recharts: https://recharts.org/en-US/
4. Chart.js: https://www.chartjs.org/
5. D3.js: https://d3js.org/

Challenge assets are available at the following GDrive location:
https://drive.google.com/drive/folders/1Dl9HavgF3Kql3iT4DInM_ZSn8Jc_36v5?usp=sharing

1. Screen mockups
    a. *Stores* page: Mockups/Stores Screen.png

    b. *SKUs* page: Mockups/SKU Screen.png
    c. *Planning* page: Mockups/Planning Screen.png
    d. *Chart* page: Mockups/Chart.png
2. Other assets
    a. GSynergy logo
    b. Sample Data

The app

1. Display a top navigation bar with the company logo on the left, and, if authentication has been implemented, a sign-in / sign-out menu to the right.
2. Display a left navigation menu with icons and label for each screen
3. The *Store* dimension screen which should allow adding, removing, and updating stores. One should be able to reorder *Store* as well.
4. The *SKU* dimension screen which should allow adding, removing, and updating *SKU*s and their *Prices* and *Costs*.
5. The *Planning* screen which shows the AG-Grid with a cross join of *Stores* and *SKUs* along the rows, and *Calendar* along the columns. The *Calendar* should group *Weeks* by *Months*. It should have the following columns for each *Week*.
    a. *Sales Units*: Editable integer values for units sold
    b. *Sales Dollars*: Non-editable calculated field, formatted as currency. It should be calculated as: *Sales Units * Price*
    c. *GM Dollars*: Non-editable calculated field, formatted as currency. It should be calculated as: *Sales Dollars – Sales Units * Cost*
    d. *GM %*: Non-editable calculated field, formatted as percentage. It should be calculated as: *GM Dollars / Sales Dollars*. The cell background should be conditionally formatted using the following rules:
        i. Green if greater than or equal to 40%
        ii. Yellow if greater than or equal to 10% but less than 40%
        iii. Orange if greater than 5% but less than 10%
        iv. Red if less than or equal to 5%
6. OPTIONAL: The *Chart* page should allow the user to select a *Store*, and show the *GM Dollars* and *GM %* in a dual axis bar chart with values along the Y-Axis and weeks along the X-Axis. To show these values for the *Store*, *GM Dollars* and *Sales Dollars* must be totaled across all *SKU*s for each *Store*, and then the totals must be used to calculate the *GM %*. Chart may be formatted in ways amenable to the charting library used.
7. OPTIONAL: Provide the capability to import the given sample data to prepopulate the application screens.
8. Page design should be responsive with a minimum width of 1080 pixels.
9. The Grid and the Chart must fit with the edges of the screen, with reasonable margin/padding.

10. Use any popular styling and/or components library, but ensure that you follow the standards of that library.

Optional: You will impress us if you

1. Include Jest based unit tests
2. Implement CI/CD to test and build the app for the main/master branch
3. Include appropriate level of logging to log errors, warning, etc.
4. Include authentication. You must provide us with demo login credentials.

Evaluation Criteria

1. Application working according to the given specs
2. Page routing
3. State Management
4. Handling of async operations
5. Performance – we recommend testing using the sample data
6. Code structuring for reusability, maintainability and testability
7. Structure of the code repository
8. Structure of code commits
9. Code comments
10. Effective communication via the readme file
11. Presentation quality in the video
12. Optional – Automated tests
13. Optional – CI/CD
14. Optional – Logging errors and warnings
15. Optional – Authentication using common providers