GSynergy React Web Challenge – A Two Page React 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 and React Native. 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.

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

The challenge should take you between 1 and 4 hours, but if you get carried away, we will not penalize you for your passion :)

General Rules and Guidance

1. Do not seek any assistance from anyone else.
2. Structure and write your code as if you are writing it for production.
3. Ensure your code is working before submitting it for review. You will not get a second chance.
4. You may use Typescript or Javascript. Choose the one that you are most proficient with. If you are equally proficient with both, we ask that you use Typescript.
5. Use create-react-app only. Do not use any other starter kits or utilities.
6. You must use a state management library such as Redux, Mobx, etc.
7. Please do 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: *GSIV20_YourFirstName_YourLastName*. 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. When done, share the link to the repo with careers@gsynergy.com.

ReactJS Challenge: Movies Browser

You will be developing a two-page web app for browsing and searching movies available at this online API: https://developers.themoviedb.org/3/getting-started/introduction.

To get an API Key:

1. Create a personal account at: https://www.themoviedb.org/account/signup
2. Once you have created an account, go to: https://www.themoviedb.org/settings/api to create an API key
    a. Usage: Personal
    b. Application Name: Interview
    c. Application URL: None
    d. Application Summary: For a developer interview project

Screen mockups are attached

1. Sketch: gsiv20_reactjs_challenge.sketch
2. png:
    a. *List* page: gsiv20_reactjs_challenge_listpage.png
    b. *Details* page: gsiv20_reactjs_challenge_detailspage.png
    c. Icons and Colors spec: gsiv20_icons_and_colors.png

The app should

1. On the *List* page, display upcoming movies, in movie cards, sorted by latest first. Each movie card should display (in one line, use ellipsis at end)
    a. Movie media (picture)
    b. Movie Title
    c. Rating (average vote)
    d. Description
2. The *List* page should allow for infinite scrolling. You may provide buttons for paging if you find that convenient (these are not in the mockup).
3. Allow searching for movies using the search API. Search results should be displayed on the *List* page itself. When search is cancelled, it should revert to showing all movies again as in #1 above.
4. When you click on a movie card, the app should navigate to the *Details* page showing movie details. Browser back button, or the home button on the *Details* page should navigate the user back to the *List* page. *Details* page should display
    a. Movie Title
    b. Rating (average vote)
    c. Year of release
    d. Length (HH:MM)
    e. Director
    f. Cast (Comma separated list of actors)
    g. Description
5. Page design should be responsive with a minimum width of 512 pixels.
6. You must use a state management library of your choice as well as common routing libraries.

Optional: You will impress us if you added automated testing

1. Jest based unit tests
2. Detox or Appium based UI tests

Evaluation Criteria

1. Design implemented according to attached UI specs using React Components
2. Routing implemented (*List*, *Detail*)
3. Connect to API - Movies retrieved, data parsed and presented in *List* and *Detail*
4. Connect to API - Search, search functionality is working
5. State Management
6. Handling of async operations

7. Performance
8. Code structuring for reusability, maintainability and testability
9. Optional – Automated tests