*xam

Senior React Developer Worksheet

Time to completion: within 24 hours

Notes:

- Please read all the instructions thoroughly.
- You are free to use Google at anytime during this worksheet.
- Code needs to follow your best practices, including what you see as essential for enterprise grade apps. (e.g. folder structure, naming standards, unit tests, etc.)
- You will be assessed based on how you implement the activities required.
    - Try to split components as much as possible.
- The take-home exam's purpose is to check how an applicant implements a feature with minimum instructions provided.
- Sample **designs and data** are provided and included in the zip file.
    - How your application looks depends entirely up to you, but it's recommended to follow the layout structure given in the design.
    - You can use any 3rd party libraries/packages if needed.
    - Use Responsive design if possible.
    - Take note that functionalities and behaviours are checked more than the design.
- Using **Redux** or **Context** is not required. But will be great if you can use it.

Activity 1 (Max. 50 pts)

1. Create a react app with **TypeScript** installed.
2. Create a simple **Login** feature.
    a. Prompts user to input **Branch ID**, **User name** and **Password** fields.
    b. Data of users will come from a mock data.
    (Please check file **users_data.ts** under the zip file)
    c. Add validation handling. (e.g. incorrect password, required checker, etc.)
        i. Points will depend on how many validation scenarios are covered.
        ii. Should not trigger any function if there is validation errors. (e.g. navigation, etc)

Activity 2 (Max. 70 pts)

1. Continue the login function from Activity 1
2. Once validation has been passed, user should be navigated to the next page.
    a. Display the username of the logged in user.
        i. How you implement the logged-in user depends entirely up to you
        (e.g. saving id in local storage, using redux, etc.)
    b. Display **LOGOUT** button
        i. Once clicked, user should be navigated back to **Login** page.
    c. Create a **Table** component that will display data from **users_data.ts**
        i. Use state management (It's up to you if you want to use redux)
        ii. Should display **Branch ID**, **Username**, **Name** and **Position** value for each user
        iii. Add an Action column that contains a **Remove** button
            1. Once clicked, target user should be removed from the list.
    d. Create an **Add User** component that can be used to add a user in the displayed list.
        i. Prompts user to input **Branch ID**, **Username**, **First Name**, **Middle Name**, **Last Name**, **Position** and **Password**.
        ii. Validations are **NOT** needed. (But will be nice if you add some.)
        iii. Display a **Reset** button
            1. Once clicked, all fields should be cleared
        iv. Display **Add** button
            1. Once clicked, input data should be added in the displayed list

Activity 3 (Max. 30 pts)

1. Write tests scripts for the **Login** feature you created in Activity 1.
    a. Cover scenarios to ensure:
        i. Items are properly rendered
        ii. Used labels are correct
        iii. Function behaviour works as expected
    b. It is recommended that you use **Jest** and/or **React-Testing-Library**.

You may submit in any of the following formats (whatever is the quickest or most convenient for you to do)

1. zip (do not include the node modules)
2. Git repository link (please make sure to push the changes)