hr | avatar
Illuminating Talent

Test Results and Interview Guide

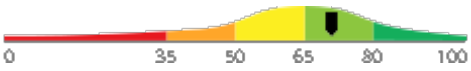
Candidate: **Elizabeth Wantsajob**
Assessment: React Programming
Completed: June 27, 2026
Prepared for: Sara Maple
Example Company

What's Included

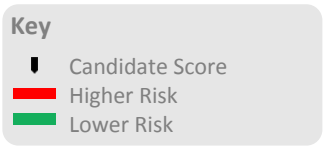
- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The React Programming assessment measures key factors related to high performance and tenure in this job. Attribute types measured vary by test, but can include cognitive ability, skills, knowledge, personality characteristics, emotional intelligence, and past behavioral history. This report includes a one page summary, followed by detailed results with an embedded interview guide. Note that these results should always be used as a part of a balanced candidate selection process that includes independent evaluation steps, such as interviews and reference checks.

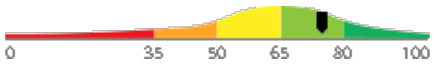
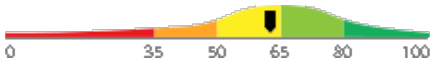
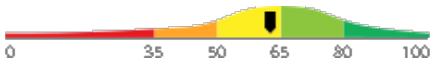
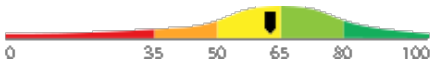
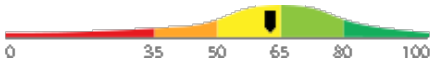
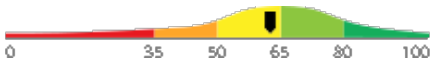
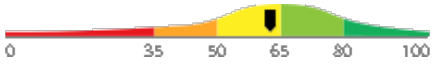
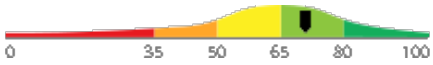
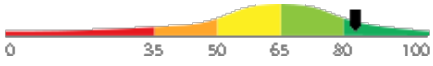
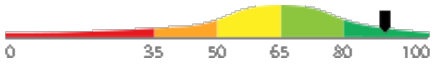
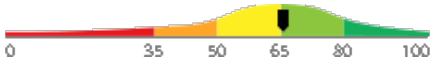
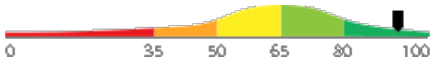
Overall

Candidate	Score	Interpretation
Elizabeth Wantsajob beth.wantsajob@gmail.com React Programming June 27, 2026	71	

The candidate demonstrates a solid and competent understanding of React development, including hooks, routing, data fetching, conditional rendering, and component lifecycle management. They are likely capable of contributing effectively to React projects with occasional need for guidance on more advanced patterns, debugging techniques, or project configuration.



Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Components, JSX, Props, and State	75	
Components, JSX, Props, and State (Coding Tasks)	62	
Data Fetching and API Integration (Coding Tasks)	62	
Debugging, Project Setup, and Code Organization (Coding Tasks)	62	
Event Handling, Forms, and List Rendering (Coding Tasks)	62	
React Hooks (Coding Tasks)	62	
Routing with React Router (Coding Tasks)	62	
Data Fetching and API Integration	71	
Debugging, Project Setup, and Code Organization	83	
Event Handling, Forms, and List Rendering	90	
React Hooks	65	
Routing with React Router	93	

Comparison

Percentile scores indicate how the candidate compares to other test-takers within various groups. The candidate scored equal to or better than the fraction of test-takers indicated by the percentile.

Test-Taker Group	Percentile
Global	71st
North America	59th
United States	59th
Example Company	65th

Artificial Intelligence (AI) Generated Scores

This table includes one or more scores derived from a large language model AI query. AI-derived scores are non-deterministic. That is, they are not precisely repeatable. Therefore, these scores should always be treated as supplementary information and should never be used exclusively or compared to hard cutoff values.

Estimated Value	Score	Confidence	Interpretation
Knowledge, Skills, and Abilities Summary	-	-	Summary Points (AI): • (Generic Text for Sample Report) Strong performer in Drag and Drop Files tasks, indicating comfort with file management and basic computer interactions. • Demonstrates solid numerical accuracy in Recognizing and Confirming Numbers, a valuable asset in detail-oriented roles. • Moderate overall performance in Analytical Thinking and Attention to Detail, with adequate grammar skills but room for improvement. • Struggles with Reading and Analyzing Problems, which may limit effectiveness in roles requiring critical reading and complex problem-solving. • Lowest performance in Navigating Between Screens, suggesting difficulty with multi-screen software workflows that could impact productivity in computer-intensive roles. Narrative (AI): Elizabeth Wantsajob demonstrates a mixed profile of knowledge, skills, and abilities across the assessed competencies. Elizabeth shows a strong aptitude in Drag and Drop Files, performing well on this technical task and suggesting she is comfortable with this type of computer interaction. This is a notable strength that would translate well into roles requiring file management and basic computer navigation tasks. In the area of Analytical Thinking and Attention to Detail, Elizabeth performs at a moderate level. She demonstrates solid ability in Recognizing and Confirming Numbers, which suggests she is careful and accurate when working with numerical data — a valuable skill in detail-oriented work environments. Her Grammar performance is adequate but leaves room for improvement, indicating she may occasionally make written communication errors. Her weakest area within this competency is Reading and Analyzing Problems, where she struggled to consistently interpret and work through written problem scenarios. This may impact her effectiveness in roles that require critical reading, written comprehension, or complex problem-solving. Elizabeth's most significant area for development is Navigating Between Screens, where she scored considerably lower than the other competencies. This suggests she may have difficulty efficiently moving through software interfaces or multi-screen workflows, which could slow productivity in roles that rely heavily on navigating computer applications or data entry systems. Overall, Elizabeth brings some useful technical strengths, particularly in file management and numerical accuracy, but would benefit from targeted development in software navigation and analytical problem-solving to be fully effective in roles that demand these skills. Computed on: April 2, 2026, 11:09:49PM EDT

Detail

Candidate:	Elizabeth Wantsajob, beth.wantsajob@gmail.com
Assessment:	React Programming
Authorized:	June 27, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	June 27, 2026, 8:11:20PM EDT
Completed:	June 27, 2026, 8:11:20PM EDT
Overall Score:	71

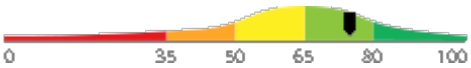
Knowledge and Skills Detail

This section contains a list of job-related knowledge areas and skills that have been evaluated. Low scores in these areas often indicate that additional learning may be required before top performance can be achieved.

Detail | Interview Guide

Components, JSX, Props, and State

Score: 75



Description:

Covers how to create functional React components, write JSX syntax, pass data between components using props, and manage local component data using state. This is the foundation of all React development and is used in virtually every React application.

Interpretation:

Candidate should achieve above average job performance in this area with little or no training.

The candidate demonstrates a solid and competent understanding of React programming, with proficiency across core topics including component architecture, hooks, state management, routing, and data fetching. This individual is likely capable of independently contributing to React-based web application development tasks with minimal oversight. Some advanced or nuanced areas of the framework may benefit from further refinement and experience.

Describe a situation where you needed to manage state in a component. How did you decide what data to put in state, and how did you update it?

1	2	3	4	5
Cannot clearly explain state or gives an example where state is unnecessary or misused.		Gives a reasonable example of useState but does not explain the decision-making process well.		Clearly explains state decisions, correct use of useState, immutability, and re-rendering behavior.

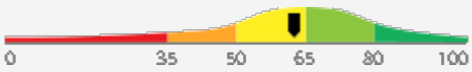
Can you explain what a React component is and describe how you would pass information from a parent component to a child component?

1	2	3	4	5
Vague or incorrect description of components; cannot explain props or confuses props with state.		Correctly describes components and props but gives a simple or incomplete example.		Clearly explains components, JSX, and props with a concrete example; mentions prop types or default props.

Detail

Components, JSX, Props, and State (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

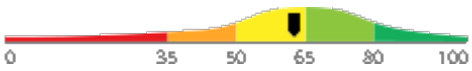
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Data Fetching and API Integration (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

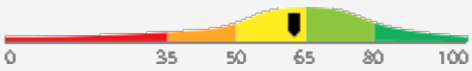
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Debugging, Project Setup, and Code Organization (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

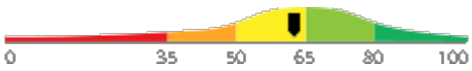
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Event Handling, Forms, and List Rendering (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

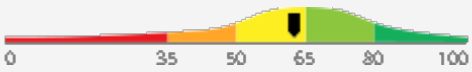
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

React Hooks (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

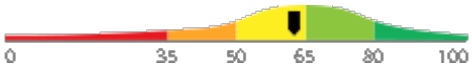
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Routing with React
Router (Coding Tasks)

Score: 62



Description:

Covers the use of pointers to reference and manipulate memory addresses, along with dynamic memory allocation and deallocation using malloc, calloc, realloc, and free. Includes pointer arithmetic, dereferencing, and avoiding common issues like memory leaks and dangling pointers.

Interpretation:

The candidate exhibits average writing skills, which can hinder high performance in some jobs.

The candidate possesses a moderate working knowledge of C programming, demonstrating familiarity with core concepts including data types, control flow, functions, and basic file I/O. They may require some guidance when working with more advanced topics such as dynamic memory allocation, modular design, or debugging complex logic.

Overall AI Score:	65.0
Lines of Code:	15.0
Syntax Errors:	5.0
AI Confidence Level:	50
Match with Ideal Response (AI):	30.0
Structure:	50.0
Syntax:	30.0

Please see below to view the essay submitted.

Interview Guide

Walk me through how you would dynamically allocate memory for an array of 10 integers, use it, and then properly release it. What issues might arise if you don't follow best practices?

1	2	3	4	5
Cannot write correct allocation code; unaware of free() or memory leak risks.		Writes mostly correct malloc/free code; identifies memory leaks but misses other risks.		Correct malloc, use, and free; identifies leaks, dangling pointers, and NULL check on allocation.

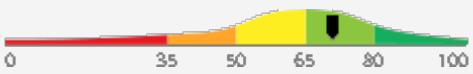
Can you explain what a pointer is in C and describe a situation where you would use one?

1	2	3	4	5
Vague or incorrect definition; cannot describe a practical use case.		Correct basic definition; gives a simple but valid use case with some gaps.		Clear definition with accurate use case; mentions address storage, dereferencing, or dynamic memory.

Detail

Data Fetching and API Integration

Score: 71



Description:

Covers how to retrieve data from external APIs inside React components using fetch or axios, typically within a useEffect hook. This includes handling loading states, errors, and displaying the returned data in the UI.

Interpretation:

Candidate should achieve above average job performance in this area with little or no training.

The candidate exhibits a solid and proficient understanding of data fetching and API integration in React, reliably implementing API calls using fetch or axios within the useEffect hook. They are generally competent in managing loading states and errors, and can effectively render API-returned data within the UI.

Interview Guide

Describe how you would handle a situation where an API request fails. What would you show the user and how would you implement that in React?

1	2	3	4	5
Does not address error handling or only mentions console.log; no user-facing feedback.		Mentions a try/catch block and setting an error state but gives an incomplete or vague UI response.		Describes try/catch, setting error state, and conditionally rendering a user-friendly error message in the UI.

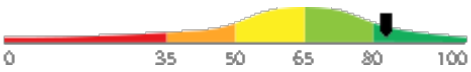
How would you fetch data from an API when a component first loads and display that data on the screen?

1	2	3	4	5
Cannot describe using useEffect for fetching or does not handle the asynchronous nature of the request.		Describes fetching inside useEffect with correct syntax but does not mention loading or error states.		Describes full pattern: useEffect on mount, async fetch, setting state with data, and handling loading and error states.

Detail

Debugging, Project Setup, and Code Organization

Score: 83



Description:

Covers how to read React error messages, use browser developer tools to inspect components and state, and set up and organize a React project using tools like Create React App or Vite with npm. This also includes understanding project file structure, configuration, and writing maintainable code.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits an advanced and comprehensive understanding of React debugging, project setup, and code organization. They are highly proficient in interpreting React error messages, leveraging developer tools, configuring projects with industry-standard tooling, and structuring codebases for long-term maintainability.

Interview Guide

How do you typically organize the files and folders in a React project, and what factors influence how you decide to split code into separate components or files?

1	2	3	4	5
Cannot describe a folder structure or places all code in one file; no criteria for splitting components.		Describes a basic folder structure with a components folder but gives vague criteria for splitting components.		Describes a clear folder structure, explains splitting by feature or reusability, and mentions maintainability and readability for teams.

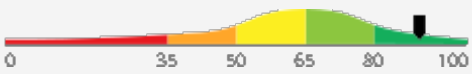
When your React application shows an error in the browser, what steps do you take to find and fix the problem?

1	2	3	4	5
Describes only refreshing the page or guessing; does not mention console, error messages, or developer tools.		Mentions checking the console and reading the error message but does not describe a systematic debugging process.		Describes reading the error stack trace, using React DevTools, checking component state and props, and isolating the issue.

Detail

Event Handling, Forms, and List Rendering

Score: 90



Description:

Covers how to handle user interactions such as clicks and input changes, manage controlled form inputs, and render dynamic lists of elements using the map method with unique keys. These patterns appear in nearly every interactive React application.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits a strong and comprehensive command of React event handling, controlled form inputs, and list rendering using the map method with unique keys. They are well-equipped to implement these essential patterns accurately and efficiently across a wide range of interactive React applications. This level of proficiency reflects a high degree of readiness to contribute effectively in roles requiring React development expertise.

Interview Guide

You need to render a list of items fetched from an API. How would you render the list in JSX, and why is the key prop important?

1	2	3	4	5
Cannot use map to render a list or does not mention keys; may use index as key without caveats.		Correctly uses map and adds a key prop but cannot fully explain why keys matter to React.		Correctly renders the list, uses a unique key, and explains how keys help React efficiently update the DOM.

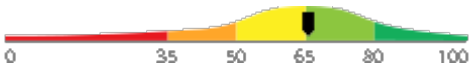
How would you create a text input in React where the displayed value is always kept in sync with your component's state?

1	2	3	4	5
Cannot describe a controlled input or does not connect the input value to state.		Describes using value and onChange correctly but cannot explain why this pattern is used.		Clearly explains controlled components, the role of onChange, and why React controls the input value.

Detail

React Hooks

Score: 65



Description:

Covers the built-in React hooks used to manage logic and data flow in functional components, including useState, useEffect, useContext, and useRef. Hooks are the primary way React developers handle side effects, shared data, and component behavior.

Interpretation:

Candidate should achieve above average job performance in this area with little or no training.

The candidate demonstrates a solid and practical understanding of React Hooks, including their application for managing state, side effects, and component behavior in functional components. They are likely capable of working independently with hooks in most development scenarios, with some room to deepen expertise in advanced use cases.

Interview Guide

Walk me through how you would use useContext to share data across multiple components without passing props manually at every level.

1	2	3	4	5
Cannot explain useContext or confuses it with props; no mention of a Provider.		Describes creating a context and using a Provider but is unclear on consuming context in child components.		Clearly explains creating context, wrapping with Provider, and consuming with useContext; mentions use cases like themes or auth.

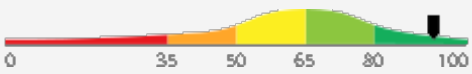
What is the useEffect hook and when would you use it in a component?

1	2	3	4	5
Cannot explain useEffect or confuses it with event handlers or lifecycle methods.		Correctly identifies useEffect for side effects but cannot explain the dependency array.		Clearly explains useEffect, dependency array behavior, and gives a practical example such as fetching data on mount.

Detail

Routing with React Router

Score: 93



Description:

Covers how to set up client-side navigation in a React application using React Router, including defining routes, linking between pages, and accessing URL parameters. Routing is essential for building any multi-page React application.

Interpretation:

Candidate should achieve superior job performance in this area with little or no training.

The candidate exhibits a strong and comprehensive mastery of React Router and client-side navigation within React applications. They are well-equipped to confidently implement and manage complex routing architectures, including defining routes, navigating between pages, and accessing URL parameters across multi-page applications.

Interview Guide

How would you create a route that accepts a dynamic value in the URL, such as a product ID, and use that value inside the component it renders?

1	2	3	4	5
Cannot describe dynamic route parameters or does not know how to access them in the component.		Mentions using a colon in the route path but is unclear on how to access the parameter with useParams.		Correctly defines a dynamic route with a parameter, uses useParams to extract the value, and describes a practical use case.

How would you set up two separate pages in a React application so that navigating to different URLs shows different components?

1	2	3	4	5
Cannot describe React Router setup or confuses routing with conditional rendering.		Mentions using React Router and Route components but cannot describe the full setup or BrowserRouter.		Correctly describes wrapping the app in BrowserRouter, defining Routes and Route components, and using Link for navigation.

IT Coding Tasks

During the assessment, the candidate was asked to write one or more programs or scripts. Their responses are included below for review.

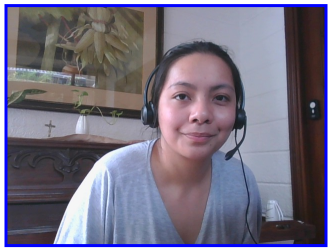
Question or Task	Response

Complete the provided partial C program by filling in the missing sections marked with TODO comments. Your completion must use standard C keywords and libraries.
 	
The program declares a function called duplicate_array that:
1. Takes a const int pointer to a source array and its length as parameters.
2. Uses calloc to allocate a new int array of the same length.
3. Returns NULL if calloc fails.
4. Copies each element from the source array into the new array using pointer arithmetic (not array subscript notation).
5. Returns the pointer to the newly allocated copy.
 	
In main, the program:
1. Declares and initializes a stack array of 4 integers with values 5, 15, 25, 35.
2. Calls duplicate_array to create a heap-allocated copy.
3. Checks for NULL and prints an error and returns 1 if the call failed.
4. Prints each element of the duplicate using a loop.
5. Frees the duplicate array.
 	
Use appropriate indentation, common C coding conventions, and add brief inline comments where needed. Type your completed source code as your response.
	#include <stdio.h>
	#include <stdlib.h>
	
	int *duplicate_array(const int *src, int length) {
	/* TODO: Use calloc to allocate a new array of 'length' integers, return
	NULL if calloc fails, copy elements from src using pointer arithmetic,
	and return the new pointer. */
	calloc(303);
	}
	
	int main(void) {
	/* TODO: Declare and initialize a stack array of 4 integers: 5, 15, 25, 35,
	then call duplicate_array and store the result. Check for NULL and
	print an error message returning 1 if it failed. */
	array[4]={5,15,25,35};
	
	int i;
	
	/* Print each element of the duplicate */
	for (i = 0; i < 4; i++) {
	printf("duplicate[%d] = %d\n", i, *(duplicate + i));
	}
	
	/* Free the duplicate array */
	free(duplicate);
	return 0;
	}
	
	Comments (AI): The code segment has several syntax errors and incomplete implementation. The duplicate_array function does not correctly allocate memory or copy elements. The main function has syntax errors and does not properly call the duplicate_array function. However, the structure and intent of the code are somewhat clear, and the code attempts to follow the requirements.

Identity Confirmation Photos

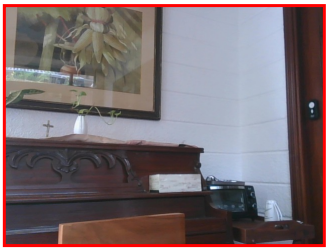
The following photos of the candidate and any identification were uploaded during the assessment session.

Photo Analysis Results	
- Risk:	Medium risk of cheating based on image inconsistencies
- Percent match among processed faces	100%
- Total images processed	17
- Total images with valid faces	14 (82%)
- Total pairs of faces compared	13
- Pairs in which faces matched	13 (100%)

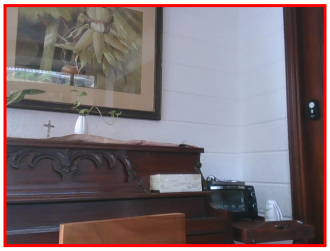


Pre/Post-Test Photo

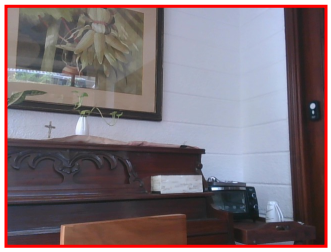
ID Photo



In-Test Error Detected (No Face Detected)

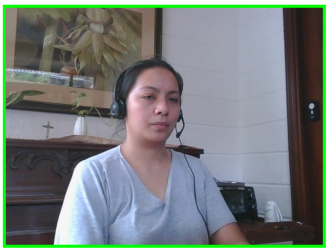


In-Test Error Detected (No Face Detected)

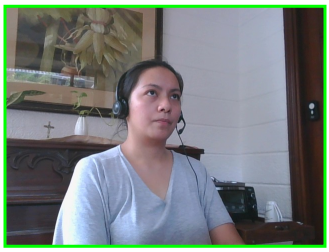


In-Test Error Detected (No Face Detected)

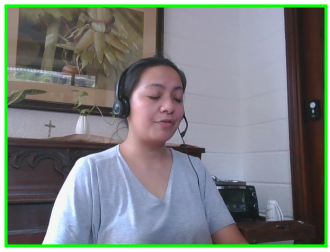
In-Test Photo



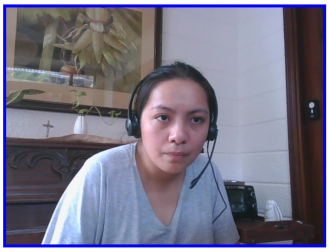
In-Test Photo



In-Test Photo



In-Test Photo



Pre/Post-Test Photo

Resume or CV

Summary Updated on

Motivated career professional with extensive experience in office administration and management. Proven track record of improving efficiency, reducing costs, and enhancing office operations through strategic initiatives and technology implementation.

Objective

I am seeking a role where I can use my many skills and my exceptional judgment and empathy for customers to make a difference to a growing company.

Education

- Associate of Applied Science in Office Administration, Portland Community College, 2020

Experience

- General Office Clerk, Paramount Office Management, 09/2023 – Present
- Administrative Assistant, Global Enterprises Inc., 04/2021 – 08/2023
- Administrative Assistant, Innovative Business Solutions Ltd., 07/2019 – 03/2021

Other Qualifications

- Microsoft Office Specialist (MOS) Certification
- Certified Administrative Professional (CAP)
- International Association of Administrative Professionals (IAAP) Certification

Report Preparation Notes

- Hiring decisions should never be based on a single source of information. The most effective use of this assessment report is as a part of a multi-faceted program of candidate evaluation that includes resume review, interviews, and reference checks.
- Overall vs Percentiles Scores: The overall score reflects the success in the test, based on the mean (average) and standard deviation of the test scores. The percentile score reflects the percentage of test-takers who scored equal or below this overall score. We recommend you use the Overall Score as your primary evaluation criteria. However, percentile scores can often be useful in comparing specific candidates against one another and with a group, such as for test takers in a certain organization or within a certain account.
- Note that comparison information is calculated based on completed instances of this assessment at that time the assessment is scored. As additional instances are completed, the comparative data may change. You can always update a report to the current values by clicking on 'Recalculate Percentiles' within the online results viewing pages at www.hravatar.com.
- Most competency scores are norm-based, which means that they can be interpreted in terms of their distance from the average or mean score. For all scales, a score equal to the mean receives a score of 65 and scores above and below this value are set so that a score change of 15 equals one standard deviation.
- For linear competencies, higher is better across the entire scale. For these scales a score between 65 and 80 (light green) represents 0 to 1 standard deviation above the mean and a score above 80 (dark green) represents more than one standard deviation above the mean. Similarly, a score of 50 - 65 (yellow) represents 0 to 1 standard deviation below the mean, while a score of 35 - 50 (orange) equates to 1 to 2 standard deviations below the mean, and a score below 35 represents more than 2 standard deviations below the mean.
- Sim ID: 20801-1, Key: 0-0, Rpt: 68, Prd: 9623, Created: 2026-06-27 20:11 EDT
- UA: Mozilla/5.0 (Windows NT 6.3; Trident/7.0; Touch; rv:11.0) like Gecko

Score Calculation Detail

The following table provides a summary of how the overall score was calculated from each of the individual competency scores. First, all competency scores are calculated on a scale of 0-100. Note that some competencies use their color category rather than their actual numeric score in the overall calculation. For these, a standard score associated with the assigned color category is used in the overall score calculation rather than the actual numeric score. This is reflected in the ''Score Value Used'' column. Next, a weighted average of scores is computed using individual competency weights, typically set using job analysis data provided by the US Government Occupational Information Network (O*Net).

Competency	Score	How applied to overall	Score Value Used	Weight (%)
Components, JSX, Props, and State	75.3696	Numeric Score	75.3696	8.3333
Components, JSX, Props, and State (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Data Fetching and API Integration	71.1445	Numeric Score	71.1445	8.3333
Data Fetching and API Integration (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Debugging, Project Setup, and Code Organization	83.2913	Numeric Score	83.2913	8.3333
Debugging, Project Setup, and Code Organization (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Event Handling, Forms, and List Rendering	90.0702	Numeric Score	90.0702	8.3333
Event Handling, Forms, and List Rendering (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
React Hooks	65.9172	Numeric Score	65.9172	8.3333
React Hooks (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Routing with React Router	93.1882	Numeric Score	93.1882	8.3333
Routing with React Router (Coding Tasks)	62.9784	Numeric Score	62.9784	8.3333
Weighted Average:				71.4043
Final Overall Score:				71

Notes

(This area is intentionally blank - it's reserved as space for your notes.)