hr | avatar
Illuminating Talent

Test Results and Interview Guide

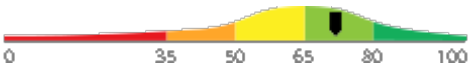
Candidate: **Elizabeth Wantsajob**
Assessment: Node.JS (Short)
Completed: June 28, 2026
Prepared for: Sara Maple
Example Company

What's Included

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

Important Note: The Node.JS (Short) 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.

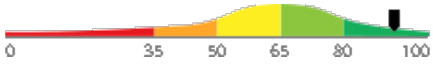
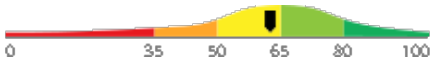
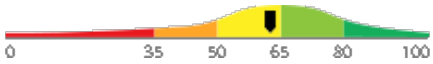
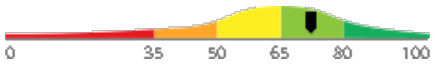
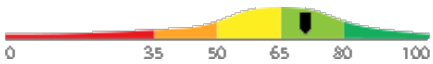
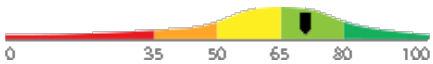
Overall

Candidate	Score	Interpretation
Elizabeth Wantsajob beth.wantsajob@gmail.com Node.JS (Short) June 28, 2026	72	

The candidate demonstrates a solid, working knowledge of Node.JS, including proficiency with asynchronous programming patterns, package management, error handling, and REST API development using common frameworks. They are likely capable of independently handling most entry-level to mid-level development responsibilities, with only occasional guidance needed for complex or specialized tasks.

Key
- Candidate Score
- Higher Risk
- Lower Risk

Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Asynchronous Programming (Callbacks, Promises, and Async/Await)	92	
Asynchronous Programming (Callbacks, Promises, and Async/Await) (Coding Tasks)	62	
Modules and Package Management (CommonJS, ES Modules, and npm) (Coding Tasks)	62	
Error Handling	72	
HTTP Servers, Express.js, and REST API Development	71	
Modules and Package Management (CommonJS, ES Modules, and npm)	71	

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	0 10 20 30 40 50 60 70 80 90 100
Global	72nd	
North America	59th	
United States	59th	
Example Company	66th	

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:	Node.JS (Short)
Authorized:	June 28, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	June 28, 2026, 12:15:31PM EDT
Completed:	June 28, 2026, 12:15:31PM EDT
Overall Score:	72

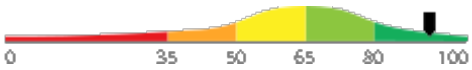
Knowledge and Skills Detail

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

Asynchronous Programming (Callbacks, Promises, and Async/Await)

Score: 92



Description:

Covers how Node.js handles non-blocking operations using callbacks, Promises, and async/await syntax. Includes understanding of the event loop, how asynchronous code is executed, and how to avoid common pitfalls like callback hell or unhandled promise rejections.

Interpretation:

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

The candidate exhibits a comprehensive and advanced mastery of Node.JS, spanning core runtime concepts, asynchronous patterns, file system operations, HTTP server management, database connectivity, testing, logging, and production deployment practices. They are well-equipped to independently lead or contribute at a high level across the full lifecycle of Node.JS application development and maintenance.

You have a function that fetches data from a database and another that fetches data from an external API, and neither depends on the other. How would you run both operations at the same time and wait for both to finish before continuing, using async/await?

1	2	3	4	5
Runs operations sequentially; does not use Promise.all or equivalent concurrent approach.		Identifies the need for concurrency and uses Promise.all correctly but omits error handling.		Uses Promise.all with async/await fluently; handles errors and explains the performance benefit clearly.

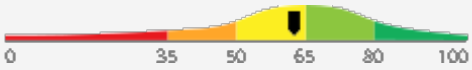
Can you explain what it means for Node.js to be 'non-blocking,' and give an example of how you would write an asynchronous function using either a callback, a Promise, or async/await?

1	2	3	4	5
Confuses synchronous and asynchronous concepts; cannot provide a working example.		Explains non-blocking at a surface level; provides a basic but mostly correct example.		Clearly explains non-blocking I/O; gives a correct, well-structured example with error handling.

Detail

Asynchronous Programming (Callbacks, Promises, and Async/Await) (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.

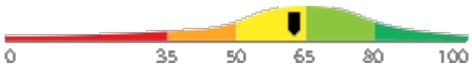
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Modules and Package Management (CommonJS, ES Modules, and npm) (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.

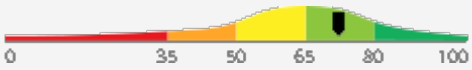
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Error Handling

Score: 72



Description:

Covers strategies for catching and managing errors in Node.js applications, including try/catch blocks for synchronous and async code, handling error events on streams or event emitters, and managing unhandled promise rejections. Includes writing code that fails gracefully and provides useful error information.

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 error handling in Node.js, including the use of try/catch for both synchronous and asynchronous code, handling error events, and managing unhandled promise rejections. They are generally capable of writing resilient code that fails gracefully and surfaces useful error information.

Interview Guide

How would you handle errors in an Express.js application so that any unexpected error in a route handler is caught and returns a consistent JSON error response to the client instead of crashing the server?

1	2	3	4	5
Unaware of Express error-handling middleware or how to pass errors to it using next(err).		Describes a centralized error-handling middleware with four parameters but may miss async error forwarding.		Correctly implements error-handling middleware, wraps async routes to forward errors, and returns structured JSON responses with appropriate status codes.

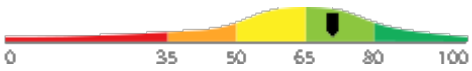
If an error occurs inside an async function, how do you make sure your application catches it instead of crashing or silently failing?

1	2	3	4	5
Cannot describe try/catch in async functions or any strategy for catching async errors.		Describes using try/catch inside an async function correctly but may miss promise rejection edge cases.		Explains try/catch in async/await, .catch() on Promises, and may mention process-level handlers for uncaught rejections.

Detail

HTTP Servers, Express.js, and REST API Development

Score: 71



Description:

Covers creating HTTP servers using Node's built-in http module and building REST APIs using Express.js. Includes defining routes, handling requests and responses, using middleware for tasks like parsing request bodies or validating input, and returning appropriate HTTP status codes.

Interpretation:

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

The candidate exhibits a solid and competent understanding of building HTTP servers and REST APIs with Express.js, including defining routes, managing middleware for tasks such as request body parsing and input validation, and applying appropriate HTTP status codes. They are likely capable of handling most practical development tasks in this area with moderate guidance.

Interview Guide

In an Express.js application, how would you create a REST API endpoint that accepts a JSON body in a POST request, validates that a required field is present, and returns an appropriate error response if it is missing?

1	2	3	4	5
Cannot set up a POST route or parse a JSON body; unaware of how to send error status codes.		Sets up the route and parses the body correctly but handles validation or error responses incompletely.		Correctly configures express.json() middleware, validates input, and returns meaningful status codes and error messages.

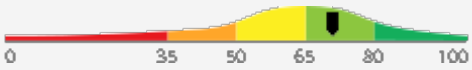
How would you create a simple web server in Node.js that responds with the text 'Hello World' when someone visits the homepage?

1	2	3	4	5
Cannot describe how to create a server; unaware of http module or Express basics.		Provides a working example using either the http module or Express but may have minor syntax errors.		Gives a clean, correct example; may contrast the http module with Express and explain when to use each.

Detail

Modules and Package Management (CommonJS, ES Modules, and npm)

Score: 71



Description:

Covers how Node.js organizes code into reusable modules using require/module.exports (CommonJS) or import/export (ES Modules). Includes using npm to install and manage third-party packages, understanding package.json, managing dependencies, and using npm scripts to automate tasks.

Interpretation:

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

The candidate shows a solid understanding of both CommonJS and ES Module systems, as well as npm workflows including package installation, dependency management, and npm scripts. They are capable of working independently on most module and package management tasks within a Node.js environment.

Interview Guide

What is the difference between dependencies and devDependencies in package.json, and how would you use an npm script to automate running your application or tests?

1	2	3	4	5
Cannot distinguish dependencies from devDependencies; unfamiliar with npm scripts.		Correctly distinguishes dependency types and can write a basic npm script entry.		Clearly explains both concepts, gives practical examples of npm scripts, and may mention environment-specific installs like npm ci.

How do you share code between two files in a Node.js project, and how would you add a third-party package to your project so you can use it in your code?

1	2	3	4	5
Cannot describe module exports or npm install; shows little awareness of how Node.js projects are structured.		Correctly describes require/module.exports or import/export and basic npm install usage.		Explains both CommonJS and ES Module syntax, describes package.json roles, and mentions devDependencies vs. dependencies.

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.

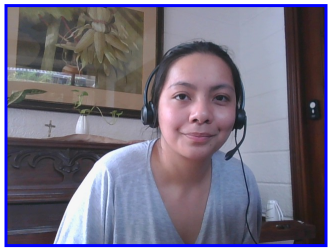
Question or Task	Response

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

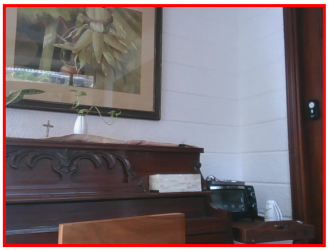
The following photos of the candidate and any identification were uploaded during the assessment session.

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%)

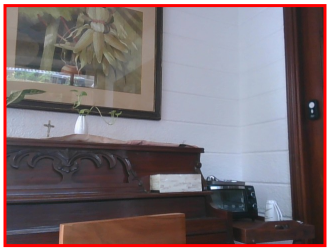


Pre/Post-Test Photo

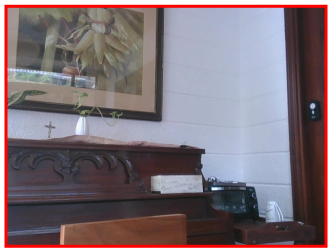
ID Photo



In-Test Error Detected (No Face Detected)

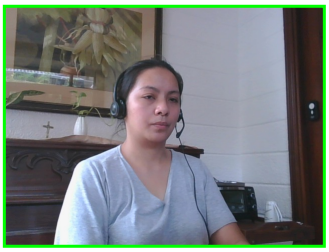


In-Test Error Detected (No Face Detected)

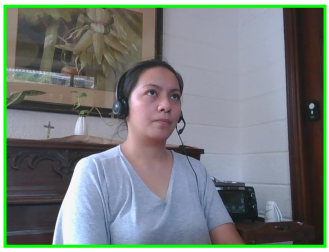


In-Test Error Detected (No Face Detected)

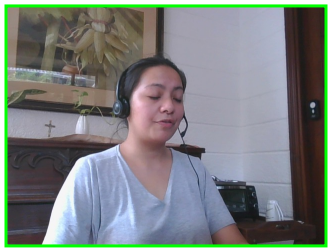
In-Test Photo



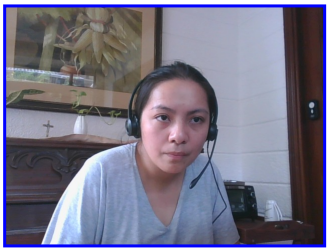
In-Test Photo



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: 20810-1, Key: 0-0, Rpt: 68, Prd: 9632, Created: 2026-06-28 12:15 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 (%)
Asynchronous Programming (Callbacks, Promises, and Async/Await)	92.2333	Numeric Score	92.2333	16.6667
Asynchronous Programming (Callbacks, Promises, and Async/Await) (Coding Tasks)	62.9784	Numeric Score	62.9784	16.6667
Error Handling	72.6942	Numeric Score	72.6942	16.6667
HTTP Servers, Express.js, and REST API Development	71.3891	Numeric Score	71.3891	16.6667
Modules and Package Management (CommonJS, ES Modules, and npm)	71.4868	Numeric Score	71.4868	16.6667
Modules and Package Management (CommonJS, ES Modules, and npm) (Coding Tasks)	62.9784	Numeric Score	62.9784	16.6667
Weighted Average:				72.2934
Final Overall Score:				72

Notes

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