hr | avatar
Illuminating Talent

Test Results and Interview Guide

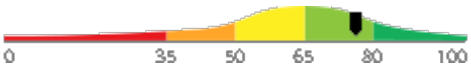
Candidate: **Elizabeth Wantsajob**
Assessment: Angular Programming
Completed: June 28, 2026
Prepared for: Sara Maple
Example Company

What's Included

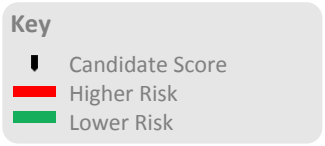
- Overall Score
- Competency Summary Table
- Comparison Matrix
- Detailed Competency Results with Interview Guide

Important Note: The Angular 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.

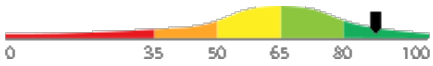
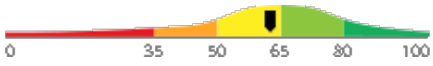
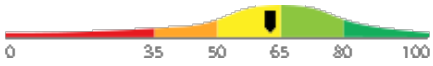
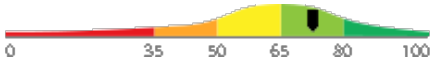
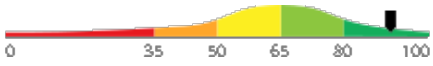
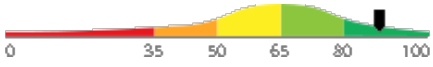
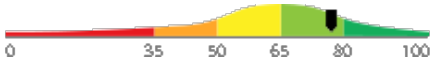
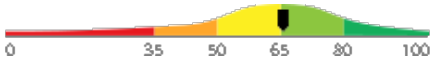
Overall

Candidate	Score	Interpretation
Elizabeth Wantsajob beth.wantsajob@gmail.com Angular Programming June 28, 2026	76	

The candidate demonstrates a solid working knowledge of Angular, including components, templates, directives, services, dependency injection, routing, and forms, and is capable of writing and maintaining Angular applications with limited supervision. Some gaps may exist in more advanced areas such as complex RxJS usage, lifecycle hook optimization, or nuanced HTTP client handling.



Competency Summary

Competency	Score	Interpretation
Skills/Knowledge (relates to immediate readiness)		
Components, Templates, and Data Binding	88	
Components, Templates, and Data Binding (Coding Tasks)	62	
Directives and Pipes (Coding Tasks)	62	
Directives and Pipes	72	
Forms and User Input Handling	91	
Observables, RxJS, and HTTP Client	88	
Routing and Navigation	77	
Services, Modules, and Dependency Injection	65	

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	76th	
North America	63rd	
United States	63rd	
Example Company	70th	

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:	Angular Programming
Authorized:	June 28, 2026, by Sara Maple, Example Company, qamailsaram.mike@hravatar.com
Started:	June 28, 2026, 1:09:41PM EDT
Completed:	June 28, 2026, 1:09:41PM EDT
Overall Score:	76

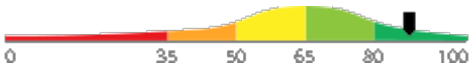
Knowledge and Skills Detail

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, Templates, and Data Binding

Score: 88



Description:

Components are the core building blocks of Angular applications. This topic covers how to create and configure components, write templates using HTML and Angular syntax, and bind data between the component class and the view using property binding, event binding, and two-way binding.

Interpretation:

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

The candidate exhibits a comprehensive and advanced understanding of Angular programming, reflecting strong proficiency across all major subject areas including components, directives, pipes, routing, reactive and template-driven forms, observables, HTTP client usage, and Angular CLI tooling. This individual demonstrates the depth of knowledge expected of a skilled mid-level or above Angular developer, capable of writing, debugging, and maintaining sophisticated Angular applications with minimal supervision. They are well-positioned to take on complex development tasks and potentially serve as a technical resource for less experienced team members.

Describe the differences between property binding, event binding, and two-way binding in Angular. When would you use each one?

1	2	3	4	5
Confuses or cannot distinguish between the three binding types; unable to give appropriate use cases.		Correctly identifies the binding types and gives basic use cases but explanation lacks depth or precision.		Clearly distinguishes all three, uses correct syntax examples, and gives practical, well-reasoned use cases for each.

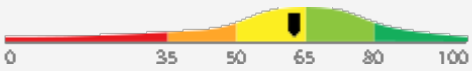
Can you explain what a component is in Angular and describe how you would display a value from your component's TypeScript class in its HTML template?

1	2	3	4	5
Vague or incorrect description of components; cannot explain how to display class data in a template.		Correctly describes a component and mentions interpolation or property binding but lacks detail or clarity.		Clearly explains components, accurately describes interpolation and binding syntax with a concrete example.

Detail

Components, Templates, and Data Binding (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.

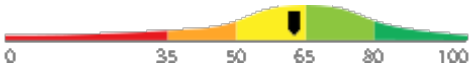
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Directives and Pipes (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.

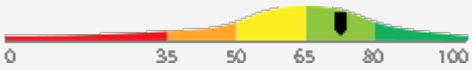
Can you explain what a pointer is in C and describe a situation where you would use one?

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

Directives and Pipes

Score: 72



Description:

Directives and pipes are used directly in Angular templates to control how content is displayed. This topic covers built-in structural directives like *ngIf and *ngFor for controlling the DOM, attribute directives for changing element appearance or behavior, and pipes for transforming and formatting data displayed in templates.

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 Angular directives and pipes, including structural directives for DOM control, attribute directives for modifying element behavior, and pipes for data transformation. They are capable of applying these concepts effectively in most practical Angular development situations.

Interview Guide

How would you use a pipe in an Angular template, and can you describe a situation where you would create a custom pipe instead of using a built-in one?

1	2	3	4	5
Cannot demonstrate pipe usage in a template or gives an incorrect explanation of when to create a custom pipe.		Correctly shows basic pipe usage in a template and gives a reasonable but generic scenario for a custom pipe.		Demonstrates confident pipe usage with syntax, gives a specific and practical use case for a custom pipe with a clear rationale.

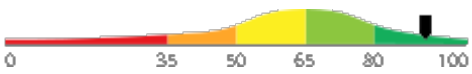
What is the purpose of *ngIf and *ngFor in an Angular template? Can you give a simple example of how you would use one of them?

1	2	3	4	5
Cannot explain the purpose of either directive or provides an incorrect example.		Correctly explains one directive and provides a basic but functional example.		Accurately explains both directives, distinguishes their purposes, and provides clear, correct examples for each.

Detail

Forms and User Input Handling

Score: 91



Description:

Angular provides two approaches for handling form input: template-driven forms and reactive forms. This topic covers how to build and validate forms using both approaches, how to bind form controls to data, and how to respond to user input and form submission events.

Interpretation:

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

The candidate demonstrates an advanced and comprehensive mastery of Angular Forms and User Input Handling. They are highly proficient in both template-driven and reactive form approaches, including complex validation, dynamic form control binding, and sophisticated handling of user input and form submission events.

Interview Guide

What are the main differences between template-driven forms and reactive forms in Angular, and what factors would lead you to choose one over the other?

1	2	3	4	5
Cannot clearly distinguish between the two form types or gives inaccurate descriptions of how each works.		Correctly identifies key differences and gives a reasonable preference but the reasoning lacks practical depth.		Clearly contrasts both approaches with accurate technical details and gives well-reasoned, practical criteria for choosing each.

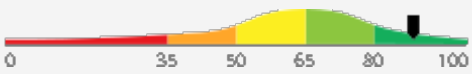
How would you create a simple form in Angular that captures a user's name and displays a message when the form is submitted?

1	2	3	4	5
Cannot describe how to bind a form input to a value or handle a form submission event in Angular.		Describes a basic approach using ngModel or a reactive form control but is missing key steps like handling the submit event.		Accurately describes a complete working approach using either template-driven or reactive forms, including binding and submission handling.

Detail

Observables, RxJS, and HTTP Client

Score: 88



Description:

Angular applications commonly use observables from the RxJS library to handle asynchronous data, such as HTTP responses and user events. This topic covers how to make HTTP requests using Angular's HttpClient, how to subscribe to and work with observables, and how to use common RxJS operators to transform or filter data streams.

Interpretation:

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

The candidate demonstrates an advanced and comprehensive mastery of Observables, RxJS, and Angular's HttpClient, including the effective use of complex operators to transform and filter data streams. They are highly proficient in managing asynchronous data within Angular applications and are well-suited to lead or contribute at a senior level in this domain.

Interview Guide

What is an observable, and how does it differ from a regular function call or a Promise? Can you describe a common RxJS operator you have used and what it does?

1	2	3	4	5
Cannot accurately explain what an observable is or meaningfully distinguish it from a Promise; cannot describe an RxJS operator.		Gives a reasonable explanation of observables and their difference from Promises and names a common operator but lacks depth.		Clearly explains observables, accurately contrasts them with Promises, and describes a specific RxJS operator with a practical use case.

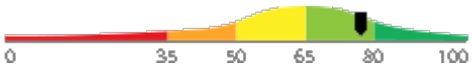
How would you use Angular's HttpClient to fetch data from an API and display the results in a component's template?

1	2	3	4	5
Cannot describe how to inject HttpClient or subscribe to an HTTP observable to retrieve and display data.		Describes injecting HttpClient and calling a get method but is unclear on subscribing or handling the response in the template.		Accurately describes injecting HttpClient, calling get with a typed response, subscribing, storing the result, and displaying it in the template.

Detail

Routing and Navigation

Score: 77



Description:

Angular's router enables navigation between different views or pages within a single-page application. This topic covers how to define routes, configure the RouterModule, use router directives like routerLink and router-outlet in templates, and pass data between routes using route parameters.

Interpretation:

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

The candidate demonstrates a solid and proficient understanding of Angular Routing and Navigation, including defining and configuring routes, applying router directives such as routerLink and router-outlet in templates, and passing data between routes. Minor gaps may exist in more advanced or nuanced routing scenarios.

Interview Guide

How would you pass a value, such as a record ID, from one route to another, and how would you read that value in the destination component?

1	2	3	4	5
Cannot describe how to define or read route parameters; confuses query params, route params, or other methods.		Correctly describes using a route parameter in the route definition and mentions ActivatedRoute but lacks implementation detail.		Accurately explains defining a parameterized route, navigating to it, and using ActivatedRoute to read the parameter with a clear example.

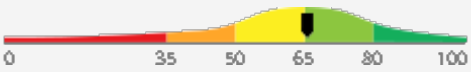
How would you set up basic navigation between two pages in an Angular application so that clicking a link loads a different view?

1	2	3	4	5
Cannot describe how to configure routes or use router directives to navigate between views.		Describes setting up routes in an array and using routerLink but omits router-outlet or key configuration details.		Accurately describes defining routes, configuring RouterModule, placing router-outlet, and using routerLink with a clear example.

Detail

Services, Modules, and Dependency Injection

Score: 65



Description:

Services are classes used to share data and logic across multiple components. This topic covers how to create services, how Angular's dependency injection system provides services to components and other classes, and how NgModules are used to organize an application and declare which components, directives, and pipes belong together.

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 Angular services, NgModules, and dependency injection. They are capable of creating and providing services, organizing application modules, and effectively managing component dependencies in most practical situations.

Interview Guide

Can you explain what the @NgModule decorator does and describe the role of its declarations, imports, and providers arrays?

1	2	3	4	5
Cannot explain the purpose of @NgModule or confuses the role of its main arrays.		Correctly explains the overall purpose of @NgModule and the role of at least two of the three main arrays.		Accurately explains @NgModule and clearly distinguishes the purpose of declarations, imports, and providers with practical examples.

What is a service in Angular, and how would you make a service available to a component that needs to use it?

1	2	3	4	5
Cannot define a service or describe how to provide or inject it into a component.		Correctly defines a service and mentions injection via the constructor but lacks detail on how to provide it.		Clearly defines a service, explains the @Injectable decorator, providedIn options, and constructor injection with confidence.

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.

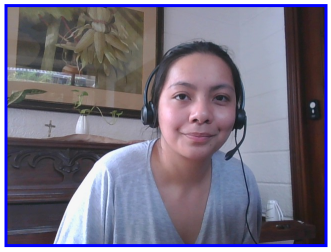
Question or Task	Response

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

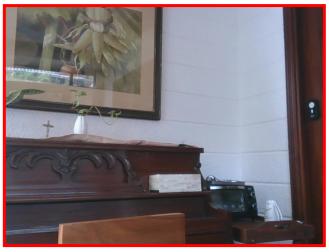
The following photos of the candidate and any identification were uploaded during the assessment session.

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%)

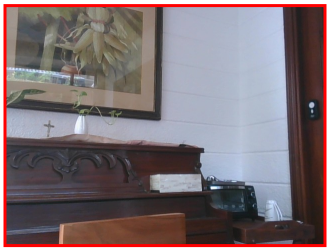


Pre/Post-Test Photo

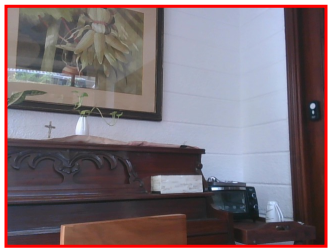
ID Photo



In-Test Error Detected (No Face Detected)

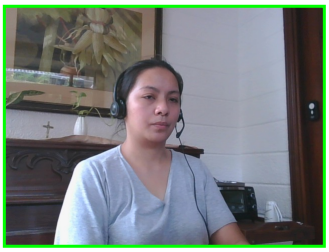


In-Test Error Detected (No Face Detected)

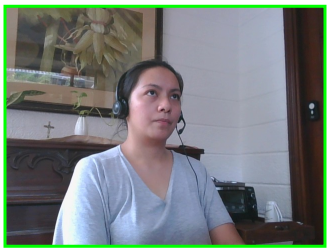


In-Test Error Detected (No Face Detected)

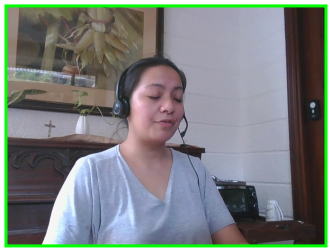
In-Test Photo



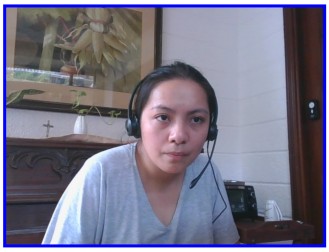
In-Test Photo



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: 20821-1, Key: 0-0, Rpt: 68, Prd: 9643, Created: 2026-06-28 13:09 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, Templates, and Data Binding	88.0362	Numeric Score	88.0362	12.5000
Components, Templates, and Data Binding (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Directives and Pipes	72.9684	Numeric Score	72.9684	12.5000
Directives and Pipes (Coding Tasks)	62.9784	Numeric Score	62.9784	12.5000
Forms and User Input Handling	91.4723	Numeric Score	91.4723	12.5000
Observables, RxJS, and HTTP Client	88.9758	Numeric Score	88.9758	12.5000
Routing and Navigation	77.5193	Numeric Score	77.5193	12.5000
Services, Modules, and Dependency Injection	65.9582	Numeric Score	65.9582	12.5000
Weighted Average:				76.3609
Final Overall Score:				76

Notes

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