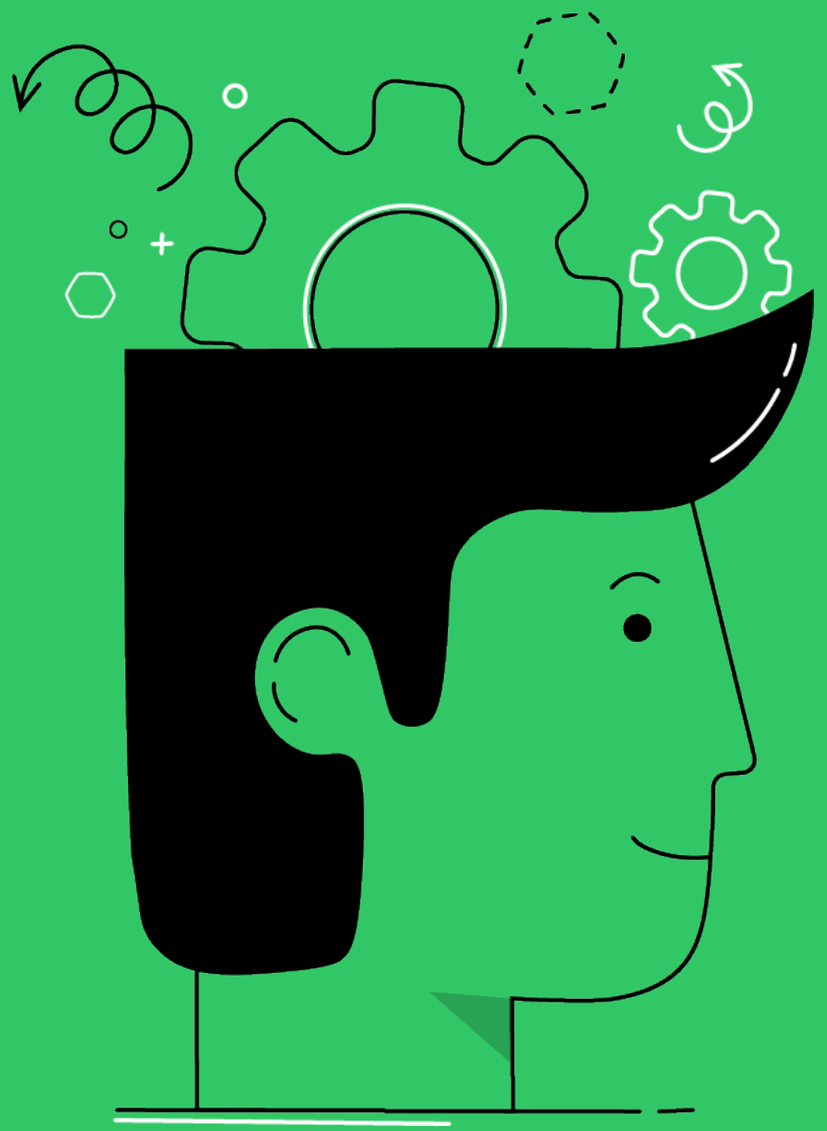
HackerRank

5 Key Skills of a New Grad Developer

www.hackerrank.com



HackerRank

What's Inside

01.

What Makes a Strong New Grad Candidate?

Hiring new grads for developer roles presents a unique challenge. Since they often have limited professional skills, assessing job fit can be a major undertaking.

When you're wading through a sea of resumes with similar work experience, degrees, and GPAs, what's the best way to seek out fit? What are the key skills that help define a high potential new grad?

We analyzed over 1,000,000 student interviews to understand how top teams assess the skills of new grads. From that data, we identified 5 key skills that define an ideal new grad candidate. In this guide, we'll explore the meaning of those 5 skills, and why they're a critical part of the new grad hiring process.

HackerRank

1,457,000
student interviews

45
countries represented

409
universities analyzed

02.

Tips for Using This Guide

This guide is intended for anyone who wants to learn more about what makes a strong new grad candidate. For the purposes of this guide, we'll focus on key skills for the most common technical new grad role: software developers.

Though the skills we describe are most desirable in entry-level software developers, the same key skills would be valuable in any new grad taking on a development-heavy role.

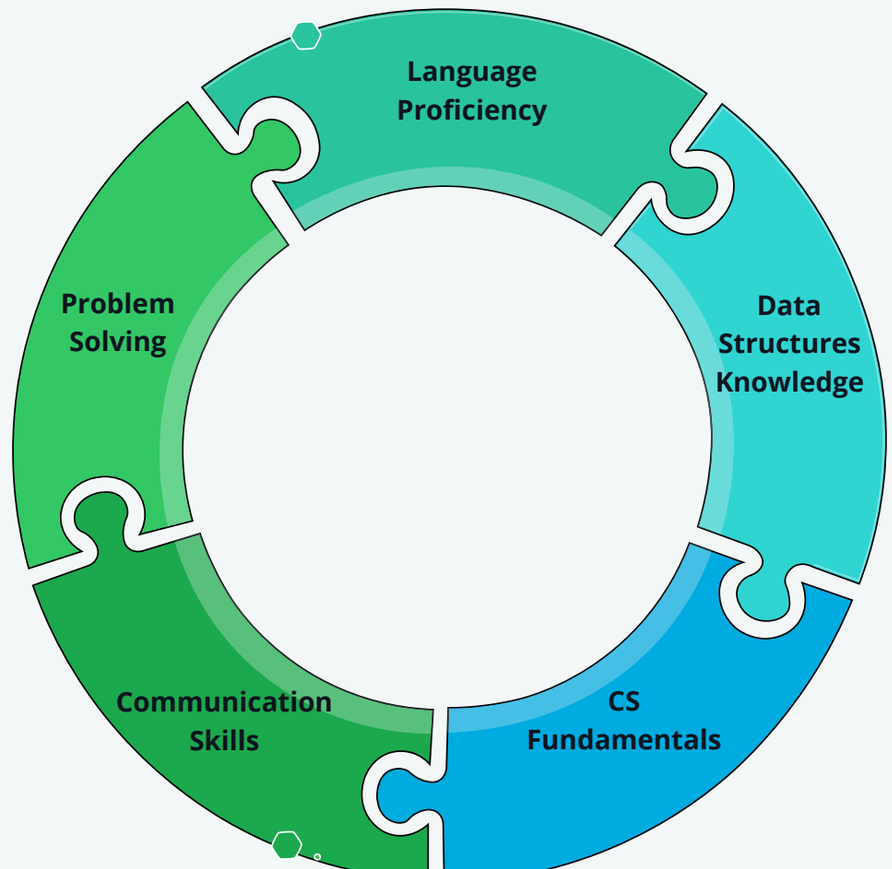
In each section, we'll explore a high-level definition of each skill. We'll also explain how the skill is used on the job, and why you should include it in your evaluation process.

03.
Key Skills of a New Grad Developer

When it comes to new grads, the best of them are strong in 5 key skills. Other key competencies—from new languages, to new frameworks, and more—can be taught over time.

Hiring new grads shouldn't be about finding an exact professional skill match, or finding a developer that has experience in your tech stack. It's about finding developers that have the foundational knowledge that will make them successful today, and the curiosity to learn new things as time goes on—which means high upside for your organization in the long term.

When it comes to software developers, however, these key skills don't exist in a hierarchy. For example: strength in problem solving doesn't make up for weakness in communication skills. They're intrinsically linked; all 5 skills are of equal importance.



HackerRank

Problem Solving

***High-Level Definition:** How a developer thinks about logical problems, and how they decide to approach them.*

How It's Utilized

Problem solving skills are defined by a developer's ability to break a problem down into its constituent components, ask relevant questions to understand the problem from multiple points of view, use their knowledge of algorithms and data structures to design a solution that fits into the time constraints provided by the problem.

As they're exposed to new projects, their knowledge base of logical problems—and corresponding solutions—will continue to grow. And so will their ability to implement them.

Why To Assess It

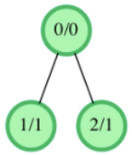
Most engineering problems take place in the context of a complex system. A developer needs to be able to break a problem down into simpler components to understand where the true issue lies. Without that ability, solving problems will be much harder. A candidate with this skill will be better equipped to solve business problems at your org—and adapt as business challenges and technologies evolve over time.

Example Problem

Optimal Virus Search

There is a virus on Alex's computer, and it must be found. Alex will search for it beginning at the root directory, folder *0*, and search through the folders in the directory tree. It takes one second to switch from one directory to its parent or a child directory, and one second to delete a virus. Determine the minimum time for Alex to delete the virus in the worst case using an optimal search method.

The directory structure is given as an array of parent directories, *parentDir*. Each index is the directory id and each value is the directory id of its parent. An array, *canBe*, is similarly structured so the index of the array is the directory id, and each value is either *0* (no virus present) or *1*, there may be a virus present. There is only *1* virus file present. In the diagrams, the folders are represented as nodes with folder id followed by *canBe* value for each folder.



For example, there are $n = 3$ directories, $parentDir = [-1, 0, 0]$. Folder *0* is the root directory and its lack of a parent is indicated as *-1*. Folders *1* and *2* have the root as their parent. The array $canBe = [0, 1, 1]$ so the virus may be in either directory *1* or *2*. Alex can traverse the directories as $0 \rightarrow 1 \rightarrow 0 \rightarrow 2$ and delete the virus from one of the subdirectories. It takes *3* seconds to traverse the directories and *1* second to delete the virus. The total time taken is $3 + 1 = 4$ seconds.

Function Description

Complete the function *minTime* in the editor below. The function must return an integer.

minTime has the following parameters:

- *parentDir:* an array of integers, *parentDir[i]* denotes the parent directory of the directory *i*
- *canBe:* an array of integers. If *canBe[i] = 1*, there may be a virus in the directory *i*, otherwise *canBe[i] = 0* and there is no virus present.

HackerRank

Language Proficiency

***High-Level Definition:** The ability to effectively express solutions through code in a given programming language.*

How It's Utilized

Once the developer has developed their solution, the next step is articulating it through code. Language proficiency means that a developer can express their solution into one or more specific coding languages (e.g. Ruby or Python).

To be proficient in a language, a developer needs to have more than basic working knowledge. They need to follow the language's rules and constructs, and demonstrate an understanding of its supporting libraries. Generally, proficiency comes from building real, practical projects with the language.

It comes down to writing code that's readable, maintainable, and makes smart use of pre-existing resources (e.g. supporting libraries). It also means writing code that performs effectively and efficiently for the use case at hand. It's important that new grad developers are proficient in at least one language and its supporting libraries.

Why To Assess It

A developer that has a proficiency in the languages you rely on can add immediate value to the team; their familiarity will help them get to know your codebase faster. But even if they aren't proficient in the languages you focus on, proficiency in another language is still an advantage. They'll have a much easier time learning new ones—which will help them keep up as technology inevitably changes.

Example Problem

☆ Vehicular Speed

In this challenge, you are required to calculate which vehicle(s) have top speed from the list of vehicles provided. Create the following classes:

1. *Vehicle*: Abstract class with the following abstract methods.
   - *int getSpeed()* method
   - *int getTopSpeed()* method
   - *String getVehicleType()* method
   - *String getVehicleName()* method

2. *Car:* This class extends the *Vehicle* class with the following variables
   - *speed:* An Integer
   - *topSpeed:* An Integer
   - *vehicleName:* A String

The class should also override the *vehicle* class methods.

- *Car(int speed, int topSpeed, String vehicleName)*
  - The method should declare a class *'variables'* with these values respectively (*speed, topSpeed, vehicleName)*
  - The method should print the following line: Vehicle {vehicleName} has been instantiated of vehicle type {vehicleType} with top speed {topSpeed}.
    - For example - "Vehicle Lamborghini has been instantiated of vehicle type CAR with top speed 200".
- *getSpeed()* : Returns the speed of the car (*km per hour*)
- *getTopSpeed()* : Returns the top speed of the car
- *getVehicleType()* : Returns "CAR"
- *getVehicleName()* : Returns *vehicleName*

3. *Bicycle:* Another class that extends the *vehicle* class with all methods and variables similar to the *Car* class with only the following change: *vehicleType:* "BICYCLE"

4. *Motorbike:* Another class that extends the *vehicle* class with all methods and variables similar to the *Car* class with only the following change: *vehicleType:* "MOTORBIKE"

5. Create a *VehicleFactory* class with the following method:
   - *Vehicle getVehicleInstance(String vehicleType, int speed, int topSpeed, String vehicleName)*
     - It accepts a string of *vehicleType* as the first argument where values can be (CAR, BIKE or MOTORBIKE)
     - Based on the *vehicleType* it returns an instance of a vehicle type with the *speed, topSpeed, vehicleName* initiated.
     - An abstract class *vehicle* can not be instantiated. It will return an instance of the child class type.

6. *static Vehicle getFastestVehicle(List<Vehicle> vehicles)*
   A static method which accepts a list of vehicles and returns the fastest vehicle based on the *topSpeed*.

HackerRank

Data Structures Knowledge

High Level Definition: *Understanding the common ways in which data can be stored, and the best way to store and retrieve it for a given use case*

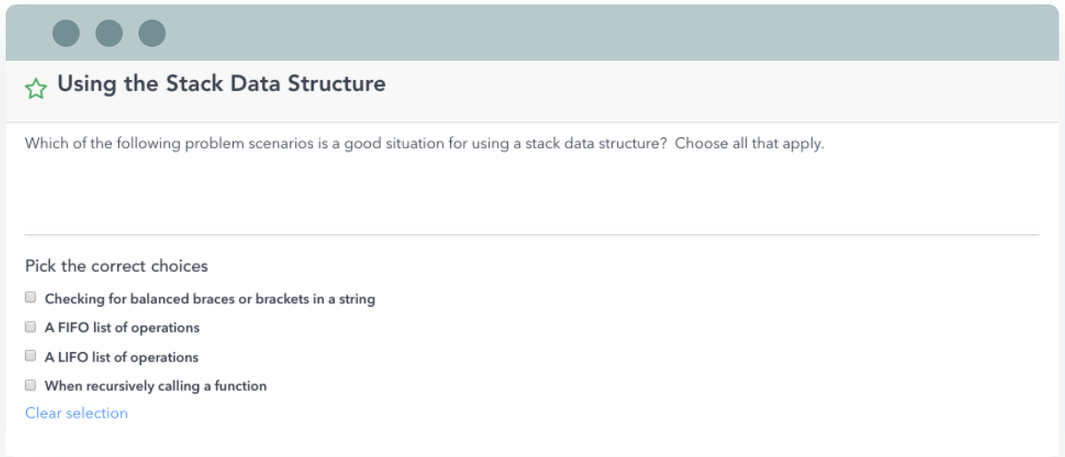
How It's Utilized

The systems built by software developers process information. But for optimal performance, that information needs to be stored efficiently. Data structures are the mechanism that store that information. To create a solution, the developer needs to know how to store information, process it, and retrieve it when needed—which is achieved via data structures. Using the right one will make storing, processing, and retrieving the information efficient at scale.

For example, when implementing an autocomplete feature, you typically use a data structure called trie. But if you used another data structure, like a linked list, your autocomplete would be significantly slower. That slowdown, in turn, could negatively impact user experience, and subsequently, harm your business.

Developers have a variety of widely adopted data structures to choose from. Though these options vary across languages, there's a common set of data structures that apply across most languages. New grads should know some of the most common ones, at the least (e.g. arrays, stacks, and more). Skill in this area is defined by knowing what data structures they can use, and knowing when to apply each of them.

Example Problem



Why To Assess It

Data structures are the building blocks of computer science. Knowledge of common data structures is crucial to learning a new codebase, and building efficient solutions. Without it, candidates risk inadvertently reinventing the wheel to address simple challenges on the job—neither a good use of time, nor a reliable approach.

HackerRank

Computer Science (CS) Fundamentals

***High-Level Definition:** A well-rounded grasp of the foundations of how computers work*

How It's Utilized

Unlike the previous 3 skills, CS fundamentals isn't limited to hands-on abilities. It encompasses a blend of hands-on skills (e.g. algorithms) and deep foundational knowledge (e.g. networking, operating systems, complexity analysis, abstraction, compilers, and more).

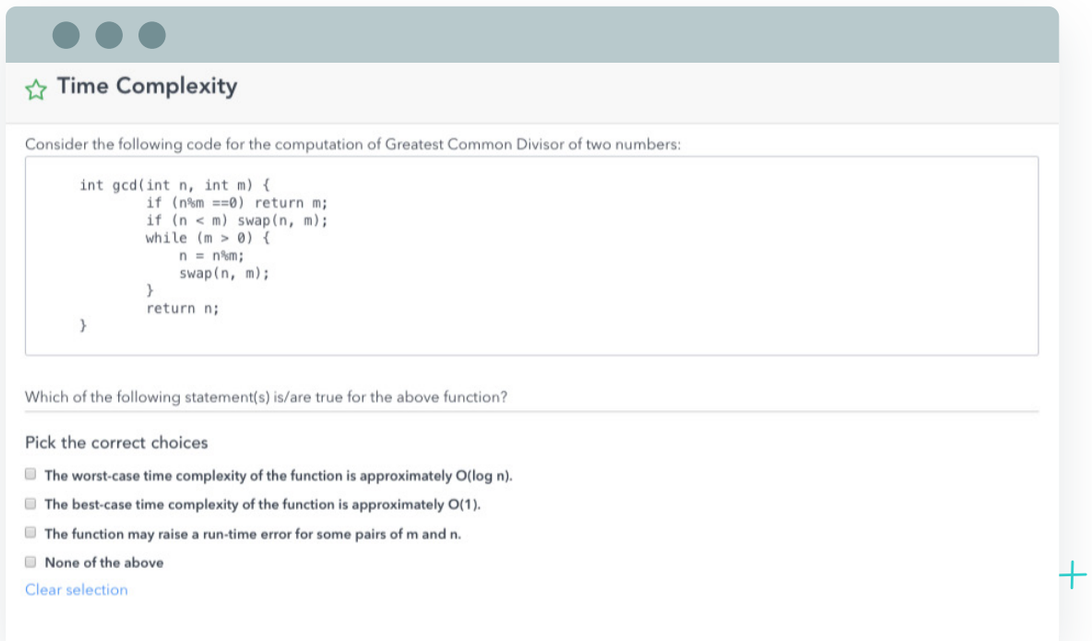
Combined, they give developers deeper context and awareness in the work: they're consciously aware of what's going on under the hood as they develop their solutions.

They go deeper than asking, "does it work?" Instead, they want to understand the solution's impact: like what it means for the system as it scales. A developer without strong CS fundamentals might be less aware of their work's broader implications—so they won't factor it into their decision making.

Why To Assess It

Having a well-rounded understanding of computer science adds depth and nuance to a developer's work. It not only helps them execute their role on a practical level, but also gives them the deep foundational knowledge they need to build reliable and performant systems.

Example Problem



Communication Skills

High-Level Definition: *The ability to collaborate with teammates in the context of the engineering org*

How It's Utilized

Even the most technically skilled developers are limited without strong communication skills.

In the context of a development role, communication skills boil down to an ability to work well with others. Since every developer's work is interconnected on a team, it's important that incoming new grads can communicate with grace: from managing expectations, to being a good listener, creating shared understanding, and more.

For better or worse, new grads often have limited work history, and subsequently, limited professional references to speak to their communication skills. Methods like pair programming and behavioral interviews will be some of your best tools to evaluate them on this front.

Why To Assess It

Unless your new grad is expected to work alone, communication skills are non-negotiable. Development is a team sport, and communication skills enable developers to collaborate with peers, managers, and other key stakeholders across your org.

HackerRank

51%

of developers say that **team collaboration** is one of their top contributors to success at work

04.

Pointers for Identifying Strong New Grad Developers

Finding strong new grads is about more than solving the challenges of your org today: it's about making a long-term investment in the future of your company. Here's what you can do to make sure you find the right fit:

- **Ensure strength in all technical skills:** Ensure strength in all technical skills: A great problem solver doesn't always have strong CS fundamentals—and vice versa. Remember that these key skills are only valuable when they're present as a collective (not individually).

- **Don't forget about communication skills:** Communication skills are sometimes pushed to the back burner. But in practice, they're just as important as technical skills. Evaluate accordingly.

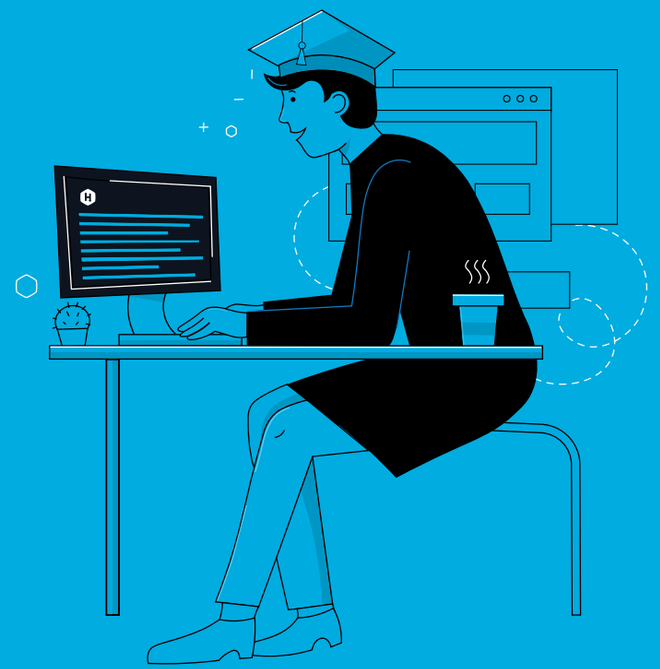
Need some inspiration for building your own new grad assessments? Learn more about how HackerRank can support your university recruiting program:

Sample Report | Request Demo

HackerRank

Methodology

Between January 2017 and June 2019, HackerRank reviewed 1,457,000 assessment attempts made by university students, including 409 universities from 45 different countries. From this data, we identified the most commonly assessed skills, and established a performance-based university ranking for different skills and languages. You can see how universities across the world performed in each skill category **here.**



HackerRank

Match Every Developer to the Right Job

HackerRank is a technology hiring platform that is the standard for assessing developer skills for over 1,500 companies around the world. By enabling tech recruiters and hiring managers to evaluate talent objectively at every stage of the recruiting process, HackerRank helps companies hire skilled developers and innovate faster.

www.HackerRank.com

USA:
+1-415-900-4023

India:
+91-888-081-1222

UK:
+44-208-004-0258

hello@hackerrank.com
www.hackerrank.com