# Technical interview questions

These examples of **Technical interview questions** can help you assess candidates' programming and engineering skills. Modify these questions for each technical interview, according to different seniority levels and positions.

## How to conduct a Technical interview

Technical interviews can be tricky, as they require specialized knowledge (e.g. of the software development process) and familiarity with related terminology. Prepare yourself before inviting candidates to an interview. Recruiters who are hiring developers and engineers should:

- Coordinate with the hiring team to identify basic technical skills candidates should have.
- Create interview questions that test whether candidates possess must-have skills required for the position.
- Ask hiring managers what to expect from candidates' answers.
- Include a written assignment that tests candidates' coding skills.

During the interview process, look for how candidates apply their theoretical knowledge on the job. Scrutinize examples from their resumes and ask for clarifications. Here are resume-based questions to consider:

- What was the project?
- Who did you work with?
- What did you develop?
- What was the outcome?

It's also important to cater your interview questions to the seniority level of each position. For entry-level positions, focus on identifying strong and weak points and potential training needs. For senior-level positions, ask candidates how much experience they have with specific tools and languages that you use.

Tech recruiters are usually familiar with Programming interview questions. However, hiring managers should ask the most complex questions, because they have better insights into their team's goals and way of working. Hiring managers can also discuss candidates' written assignments with them, provide feedback and ask follow-up questions.

## Example Technical interview questions to ask candidates

**For entry-level roles**

- What programming languages are you most familiar with?
- Describe the troubleshooting process you'd follow for a crashing program.
- How can you debug a program while it's being used?
- What is your field of expertise and what would you like to learn more about?

**For senior-level roles**

- Have you implemented significant improvements to an IT infrastructure? What were they, and how did you implement them?
- What's the most effective way to gather user and system requirements?
- Describe a time you had to explain technical details to a non-technical audience. How did you modify your presentation?
- Where do you place most of your focus when reviewing somebody else's code?

**Assignment review**

- What would you have done differently if you had more time?
- What would you do differently if you were under a strict deadline and you couldn't meet the project scope? Which features would you prioritize?
- What did you find most challenging about this assignment? What resources did you use to complete the assignment?

**Resume review**

- In which of your previous positions/past projects did you use [X] software?
- Tell me about [X] project. Who did you work with and what was your specific contribution? Describe the timeframe and how you worked within it.
- What did you learn from [X] project?

## Interviewing tips for technical roles

- Computer Science is an evergreen discipline. Keep an eye out for candidates who enjoy following trends and learning. Potential hires who test new software, participate in coding meetups and are active on technical forums and blogs are invested in their industry.
- Brainteasers and trick questions don't reveal candidates' skills. Be specific. Ask about candidates' experience with software you use and how they would approach a relevant problem likely to arise in their position. These types of questions will also help you compare candidates' answers.
- Too many theoretical questions (like "Give me the definition of…") can get tiring. Also, they don't measure candidates' problem-solving abilities. Include situational and behavioral interview questions that show how candidates perform in real-life projects.
- A written assignment should follow a first screening, usually by phone. Inform candidates

about the written assignment and email them detailed instructions. Give them enough time to complete the project, and make sure you are clear about the deadline.
- When evaluating the assignment, avoid focusing only on the right or wrong answers. Gauge candidates' way of thinking. An innovative, out-of-the-box solution (even if it's not error-free) can reveal a creative mindset needed for the role.

## Red flags

- **Unclear answers.** Candidates who struggle to explain their resume might have had little or no participation in the projects they listed. Ask follow-up questions to identify their exact roles and contributions.
- **Lack of energy.** Developers are passionate about their profession, even if you can't tell at first sight. Ask candidates about fun side projects, or about their favorite tools. Their reactions can indicate how committed they are to the field.
- **Inflexibility.** You can't expect candidates to know every software or framework that you use. But, candidates who are unwilling to adjust to your way of working are less likely to collaborate with your team. Opt for candidates who showcase a desire to learn and aren't discouraged by getting used to new systems.
- **Bad team players.** Developers might usually work in front of a computer screen, but they need to communicate with various people and teams on a daily basis. Poor interpersonal skills and signs of rudeness or arrogance indicate lack of team spirit.
- **Order-takers.** Candidates who fail to see the "big picture" are not able to understand your company's needs and objectives. Consider candidates who engage in the full software development life cycle. These people are proactive and suggest solutions – they don't simply wait for instructions.