Build
for
everyone

Candidate Interview Guide
Application Engineer, Generalist

Time to Prepare!

Congratulations on making it to the next steps of the Google interview process!

This will comprise of (5) 45-60 minute interviews, done over Google Hangouts, meeting with members from our Application Engineering teams. In order to best prepare for these technical interviews, please review the information below and reach out to your Google Staffing contact if you have any questions!

Now, let's talk prep!

Please make sure to be in front of a computer with the Google Docs open the entire time. It is a live document and will be used for any kind of technical questions. Think of this document as a virtual whiteboard - *please note*. This live doc is a shared space where you and the interviewer can exchange ideas/questions/answers visually. The interviewer can see where your cursor is, what you copy/paste, and you can make edits to what they are typing and vice versa. Use the doc to show all your work. Please make sure to check that you are able to access/type in it at least 48 hours before the interview. If you are having problems, let your Staffing contact or Recruiting Coordinator know!

- Please note that you will lose *access* to the doc after the interview.

In regards to the review and conversation surrounding your experience/resume. Focus on the details and scope of the projects and responsibilities surrounding your work and how they have evolved over the course of your career. We are interested in getting to know more about your unique skills and accomplishments. Check out **How We Hire**, **Life at Google**, **How to Prepare for our Technical Interview**, and **Example of a Coding Interview.** Additional preparation link(s): **All you need to know about the interview process at Google.**

Most importantly - you've made it this far! This is a great achievement as GOOGLE seeks the best talent in the world. Aside from your technical skills - the interviewers want to see your energy, enthusiasm and passion shine through on your interview. We can't reiterate this enough!

During the Interview

- **Think out loud.** We are interested in your analytical skills and thought process. Sometimes we are more interested in how you come to a conclusion rather than the answer itself.
- **Organize your thoughts:** Try not to leave too much silence when you are thinking. Let the interviewer know what is going on in your head, but keep your communication clear.
- **Presentation of your answers is key:** Since people are subject matter experts in their field, they often get too excited and throw out all of their knowledge about a topic.

Points to Keep in Mind

- Understanding the question is extremely important and we encourage you to ask clarifying and relevant questions before jumping into your solution
- We want to see you think through all scenarios and ask questions upfront, calling out any assumptions you are making
- Take a methodical and deliberate approach and ensure you are the interviewer are on the same page

Focus Areas

- (A) Coding & Programming x 2
- (B) Application Design & Domain Knowledge
- (C) System Integration
- (D) Googleyness & Leadership
- (5) Total Interviews
- Familiarize yourself with Google Docs and Google Drawings

(A) Coding/Programming

- Fluency in common data structures for Java or Python (Avoid flipping between programming languages mid interview)
- Most questions are basically iterating over a string or an array or some other structure that's not much more complex (eg 2d array, tree)
- Fairly complex algorithm type of questions: arrays, trees, char manipulation, link list, (Basic algorithm type of problem: given a string of variables, find all variables less than or equal to 2)
- **Trees**: Know about trees; basic tree construction, traversal and manipulation algorithms. Familiarize yourself with binary trees, n-ary trees, and trie-trees. Be familiar with at least one type of balanced binary tree, whether it's a red/black tree, a splay tree

or an AVL tree, and know how it's implemented. Understand tree traversal algorithms: BFS and DFS, and know the difference between inorder, postorder and preorder.

- **Algorithm Complexity, Design, & Analysis:** It's fairly critical that you understand big-O complexity analysis. Again run some practice problems to get this down in application. Sample topics: big-O analysis, sorting and hashing, handling obscenely large amounts of data.
- **Sorting:** Know how to sort. Don't do bubble-sort. You should know the details of at least one n*log(n) sorting algorithm, preferably two (say, quicksort and merge sort). Merge sort can be highly useful in situations where quicksort is impractical, so take a look at it.
- **Hashtables:** Arguably the single most important data structure known to mankind. You absolutely should know how they work. Be able to implement one using only arrays in your favorite language, in about the space of one interview.
- **SQL:** Simple or fairly complex queries (queries that can be solved with joints, refresh on your select statements, when to use what, how to use performance implements) - If you are not comfortable answering SQL questions, please let your Recruiter know.
- **Evaluation:**
    - Ability to produce neat and clean code
    - Efficiency in code and solution
    - Ability to exhibit good code hygiene
    - Correct syntax
    - Check for corner cases or edge cases
    - Test your code
    - The interviewers want to see your true syntax code as there will not be a compiler/ID available

(B) Application Design & Domain Knowledge

(We will ask a single application design question that will cover strategies in general):

- Ability to come up with flexible design from front-end to back-end systems using frameworks and tools and showcasing your ability to translate requirements into an application design
- Ability to walk the interviewer through all of the elements of the application that would have to exist and be pre-defined before APIs can work
- **Demonstrate an ability to:**
    - Parse requirements
    - Propose meaningful functionality
    - Design a neat and clean data model
    - Specify in detail the concepts of the system architecture
        - Security (internal/external)
        - Users/Audience
        - Reliability (Consistency of response)
        - Recovery (Crash in-between and restart from last transactions)

        - Monitoring/Logging (In case of problems occurring at different APIs)
- Demonstrated experience in scaling the systems for performance for both database and business suites
- Familiarity with general concepts of ERP architecture, Object Oriented and data modeling and transactional systems
- **Examples of System Design Question** (Design an Airline Ticket Booking system with Data Model, Design a Library Management System with Data Model)
- Some common mistakes in the Design interview include:
    - **Immediately jumping into a solution**
        - Why this is bad:
        - We have no idea if you are going down a rabbit hole
        - We may not be able to offer advice of hints if you get stuck
        - You'll waste time if you decide to start over with a new approach
        - Design questions mimic the ambiguity of the real world
    - **Not running through an example**
        - DO pick a simple example
        - DO think about an edge case or two
        - DO be clear with your example input (overly specific vs. vague)
        - DON'T waste too much time running through many examples
    - **Not providing technical details**
        - High-level design is ok to help formulate and communicate your initial solution
        - DO use open-source or industry standard platforms as part of your solution (think: components generally available in Google Cloud Platform)
        - DON'T propose "black-box" solutions
        - Forgetting about the problem you are solving for
        - DO support your solution; justify your recommendation with arguments
        - DO highlight trade-offs and decision points
        - DO discuss the design choices you are making
        - DON'T go off on a tangent or spend too much time on one small detail

Domain Knowledge:

- This is an opportunity to showcase the depth of your business domain knowledge and demonstrate experience with and understanding of current/past employers' business processes
- This part of the interview will feel more functional than it is technical
- We are evaluating your ability to model a business process and map it to technology solutions.
- **Example Question:** Describe a business process you've recently been involved in? (We want to see your ability to describe the overall business process it was a part of, the specific functionality of your solution, and how it fits into the overall whole)
- Ability to understand new business processes/problems and identify the requirements for which you would configure current implementation, customize/extend standard

implementation or integrate with other third party/ home-grown applications.

(C) System Integration Knowledge/Ability

(NOTE: We will ask a single system integration question that will cover integration strategies in general):

- Experience and knowledge of integrating 3rd party systems, process, constraints, benefits, etc.
- Demonstrate your ability to build out middleware implementations and solutions
- Cover the integration flow both inbound and outbound and discuss:
    - Security, networking, web service authorization/authentication, protocols (SSL)
    - Think through: SLA's, technology considerations, interoperability and scalability.
- Be ready to discuss integration technologies, web standards, interoperability, protocols, security, middleware, etc.
- **Example:** How do you get System A to talk with System B so that they work seamlessly with other internal applications, such as expense reporting or payroll

(D) Googleyness & Leadership

Other Interview Tips:

Substantiate
Make sure that you substantiate what your CV/resume says – for instance, if you have Java or Python on your resume, be prepared to answer questions using these languages.

Explain
We want to understand how you think, so explain your thought process and decision making throughout the interview. Remember, our engineers are not only evaluating your technical abilities, but also how you approach problems and how you try to solve them. Explicitly state and check assumptions with your interviewer as you solve a problem to ensure they are reasonable.

Clarify
Ask clarifying questions if you do not understand the problem or need more information. Many of the questions asked in an interview are deliberately underspecified because our engineers are looking to see how you engage the problem. In particular, they are looking to see which areas you think are the most important when solving a technological problem.

Improve
Think about ways to improve the solution that you present. In many cases, the first solution that springs to mind isn't the most elegant and may need some refining. It's worthwhile to talk

through your initial thoughts with the interviewer. Jumping immediately into presenting a brute force solution is a great start, but do take time to explore a more efficient solution.

Ask Questions
At the end of the interview, most interviewers will ask you if you have any questions about the company, work environment, their experience, etc. This is your chance to learn more about the role, the projects, and the type of work you'll be doing at Google.

INTERVIEWING PITFALLS TO AVOID

- When candidates jump into design or coding without first analyzing the problem or asking clarifying questions.
- When candidates don't talk out loud. Practice speaking out loud through your thought process.
- When candidates don't pick up on hints or when candidates give up on a problem. The interviewer is NOT trying to trip you up, any hints suggested are intended to get you back on track.
- Ensure you are able to program the solution you are suggesting. Avoid suggesting an algorithm than being unable to produce the code.

LAST FEW TIPS

- If you don't understand something, ask questions!
- Please keep in mind that everything on your resume is fair game for the interviewer to inquire about. Understanding the trajectory of your career aids our interviewers in their overall assessment.
- Think through and rationalize your answer. Always optimize!
- Ask clarifying questions before formulating a response! Example: How many passengers plan on taking this car on a daily basis? This impacts if I need 2 or 4 doors. What kind of weather are you planning on driving the car in? This impacts if I need all wheel drive or not.
- If you need to make an assumption, explain why you are making it in regards to your solution.
- Let your personality shine! Show you are excited to possibly be a part of an innovative and creative company. Read about some new cool things Google is doing and familiarize yourself with our company culture and founders.

AFTERWARDS

Your Staffing partner will reach back out to you to discuss next steps as soon as they receive

the interview feedback. Please allow a few days after the interview date to expect feedback. We will keep you updated if we experience latency issues for any reason. Please do not hesitate to reach out to your Staffing partner with any additional questions/issues. We very much appreciate the time and effort you have put into preparation and with your interest in Google.

Good luck!

The topics below are trends that are seen in interviews. Nothing is a guarantee to be asked. These pointers are intended to give you a broad idea of some things that could pop up. Sometimes the ones not covered now, are discussed in later interviews if you are approved to move to the next step.

Possible questions regarding:

- Your background: Provide a clear and concise summary. Try not to drag it on, but give the interviewer a relevant picture.
- Object Oriented coding on the whiteboard (please refer to the job description).
- Basic algorithm type questions (arrays, trees, char manipulation, linked lists, O(N) analysis, etc.) and be able to do them fast.
    - **https://www.geeksforgeeks.org/fundamentals-of-algorithms/**
    - **https://www.geeksforgeeks.org/data-structures/**
    - **http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=alg_index**
- SQL questions: Ask to skip SQL questions if you are not comfortable with them (people who say they know SQL should be very strong at it. This includes grouping, subqueries, ordering, etc.).
    - **https://www.hackerrank.com/domains/sql**
    - **https://www.w3schools.com/sql/sql_exercises.asp**
    - **https://www.w3resource.com/sql-exercises/**