Software Developer - Preparation Document

Dear Students,

We hire tech enthusiasts with a broad set of technical skills who are ready to tackle some of technology's greatest challenges. The hiring process has been designed from the ground level to avoid any false positives and in order to help you to get through our process.

We have curated this document after examining some of the most frequently asked questions & also keeping in mind the preparation that you may require in order to crack our selection process. This document will come in handy in order to understand the position and the tips and tricks that will help you prepare for the hiring process for Software Developer at Josh Technology Group.

Please Note:- This document is intended to provide you with the required guidance and sample material that would be helpful in the preparation and this in no way guarantees your selection.

Excited much to participate in the selection process? We look forward to your participation!

Wish you all the luck for the hiring process!

Let's Get Started

How to Prepare :

1) **Start your prep early.** Before your interview date, set aside a specific amount of time every day to practice your coding, algorithmic, and problem-solving skills.
2) **Do as many coding questions as you can.** Visit Glassdoor, CareerCup, Project Euler, or another site listed in the appendix of this guide. The idea isn't to solve every question but to become familiar with the pattern of interpreting a question, formulating a solution, and writing an efficient, bug-free program without a compiler.
3) **Write code in a simple text editor.** In the interview, you'll write your code in a similar environment without autocompletion and code execution support.
4) **Ramp up your speed.** Time yourself. Practice under time pressure: coding speed is important. To prepare in hard mode, practice with a colleague playing the worst interviewer ever. The more rigorous your training, the easier you'll find the interviews. Try to complete two coding problems in 30-40 minutes. You can also practice by setting harsh time limits. If you're able to solve the medium-level questions within 20 – 30 minutes, it's usually a good sign.
5) **Go over data structures, algorithms, and complexity.** Be able to discuss the Big O complexity of your approaches. Don't forget to brush up on your data structures like linked lists, arrays, stacks, queues, and trees. Also sorts, searches, and traversals (BFS, DFS). And review recursion and iterative approaches.
6) **Resources Coding questions and exercises**
    a) To study runtime complexity: Big O Cheat Sheet
    b) Gauge how prepared you are on CS fundamentals with more than 100 Questions that will take from less than a minute to about an hour to solve.
    c) Engineer favorites for practicing coding problems:
        i) LeetCode
        ii) HiredInTech
        iii) HackerRank
        iv) CareerCup
        v) CodeChef
        vi) Project Euler
        vii) GeeksforGeeks

Considerations that lead to a positive outcome:

Interviewers will weigh the success of an interview based on the approach as much as the answer. They'll funnel your performance based on the following considerations:

1) Do you listen carefully and comprehend the question?
2) Do you ask the correct questions before proceeding?
3) Do you notice and follow hints the interviewer gives?
4) Are you quick to comprehend/solve problems?
5) Do you enjoy finding multiple solutions before selecting the best one?
6) Do you keep seeking out new methods to tackle the problem?
7) Are you inventive and flexible in your solutions, and open to new ideas?
8) Do your questions lead to more complex problem-solving?

Some Additional Tips :

1) Avoid misunderstanding. When we ask you to provide a solution, first define and develop a framework of the problem as you see it. Ask for help or clarification, and spend 2 – 5 minutes asking the interviewer about corner cases on the problem. This will ensure that you've understood the problem correctly.
2) Think out loud. It helps your interviewer follow along, learn about your problem-solving skills, and provide hints if needed. Discuss initial ideas and solutions, which will help you to clarify any ambiguity.
3) Avoid solutions with lots of edge cases. Avoid solutions with lots of edge cases or huge if-else blocks. Deciding between iteration and recursion is always an important step.
4) Write a working solution and iterate. It's better to have a non-optimal but working solution than random fragments of an optimal but unfinished solution.
5) Hints. If your interviewer gives you hints to improve your code, please use them.
6) If your solution is getting messy, step back. Most coding interview questions are designed to have reasonably elegant solutions. If you have if-else blocks and special cases everywhere, you might be able to take a better approach. Look for patterns and try to generalize.
7) Plan your approach. Ensure that you spend time planning your approach, but remember you can always go brute force and then optimize from there.
8) Clarifying questions. Make sure you're asking clarifying questions as you go along (ensure you have all the info you need).