**Virtual Onsite Prep – SDEII**

**Interview Structure**

An interactive deep dive into the SDE interview: <https://www.amazon.jobs/en/software-development-interview-prep#/>

Additional self-guided prep: <https://www.amazon.jobs/en/landing_pages/software-development-topics>

Very thorough external article “**Ace the coding interview, every time”:** <https://medium.com/@nick.ciubotariu/ace-the-coding-interview-every-time-d169ce1fd3fc>

*You will meet with 4- 6 Engineers in 1-hour sessions where the will each ask 1 Technical problem and 2 behavioral questions.*

* *First 5 minutes – Introduce yourself & build rapport*
* *Next 25 – 30 minutes - 1 Technical Question (System Design & 3 Coding Questions)*
* *Next 20 minutes - 2 Behavioral Questions designed to see if you exude the leadership principles*
* *Last 10 minutes - reserved for you to ask questions (time permitting) so have a few ready.*

**Leadership Principles -** [**http://www.amazon.jobs/principles**](http://www.amazon.jobs/principles)

*Most successful candidates write out at least 10 scenarios in the STAR Format before their interview. Here is an example of telling a story in the STAR format:*

* The Situation was, 50% of my customers were experiencing an issue.
* *I assigned myself the task to solve this issue for the customer.*
* *The actions I took included: A, B & C – Then X, Y, & Z.*
* *The Result was, I was able to solve this issue for 90% of the customers within a two-week time frame and 95% of those customers never experienced the issue again.*
* Make sure you include measurable metrics in the Situation and Results sections of your scenarios!
* I like to add an extra R to STAR for Reflect
* *Your interviewers will ask you follow up questions, they are trying to see if you can be self-critical.*
* *When we ask about past mistakes, we are not trying to corner you or find fault. We want to know if you learned from the situation, can you take ownership and how you put mechanisms in place to avoid it in the future.*
* Most candidates spend too much time in the situation and task section and not enough on the Action and Result Section. • Situation is just enough to put the interviewer in your shoes
* *Task is typically one sentence. Even large tasks can be summed up simply in one sentence. This is where writing out your scenarios ahead of time will help.*
* *Action – This should be your lengthy section. Talk about all the actions you took to solve the problem and/or go above and beyond for the customer. Be specific about what You did.*
* *Result – Tie a nice bow on the end of your story. Lead with measurable metrics. You want the majority if not all of your stories to have a positive result that is supported by data.*
* *My advice is to have 2 impressive, data driven examples for each of the following leadership principles:* 
  + Customer Obsession
  + Ownership
  + Invent and Simplify
  + Are Right, A Lot
  + Learn and Be Curious
  + Insist on the Highest Standards
  + Think Big
  + Bias for Action
  + Earns Trust
  + Dive Deep
  + Have Backbone; Disagree and Commit
  + Deliver Results

*Here are some final tips to consider as you write and practice your answers to behavioral questions:*

* *Did you mention the customer? Amazon is customer obsessed and it is highly important you include the customer, whether internal or external in your situation. Include the customer's viewpoint by starting with the customer working backwards.*
* *Stay positive*
* *Say I not we – we want to know what you did*
* *Frame your stories in* ***Measurable Metrics - Quantify!*** *Provide data for examples whenever possible. Mentioning this again because we are a data driven company and your interviewers will looking for the numbers.*

I also highly recommend taking a look at this LinkedIn article. It is written by a Senior Leader at Amazon, and it gives some great insight into our how to interview at Amazon, relating specifically to our LPs. Check it out: <https://www.linkedin.com/pulse/how-interview-amazon-leadership-david-anderson>

**Technical Competencies**

**Criteria Code**

*Your code will be evaluated on 3 areas.*

*1. Problem Solving – Were you able to solve the problem correctly? Did you come up with the Optimal Solution (No Brute Force)*

*2. Logical and Maintainable – Is you code Logical & Maintainable? Is it Extensible? Could another engineer come in a read your code without you being present*

*3. Production Ready - How Close to Production Ready is the Code? You should aim for no errors*

**Step Coding Methodology**

*Let’s review this coding methodology designed to help you reach the optimal solution within 25 minutes:*

*1. Clarify the Question - ask clarifying questions to make sure you understand all the variables and address any ambiguity – Ideal time - 2 minutes*

*2. Rephrase the Question - Repeat the question back in your own words so the interviewer knows you understand the problem, highlight the signature, the inputs you are given and what type of method or function you will be using – Ideal time 1 minute*

*3. Layout the Algorithm - In order to save time and optimize communication write down your algorithm either before or as you review it with your interviewer. Make sure you state time complexity and have your interviewer agree – Ideal time 3 - 4 minutes*

*4. Write your code — I have found it is hard to talk and code so for best performance let the interviewer know you are going to focus on Coding for the next 10-15 minutes.*

*5. Review your code - Optimize, validate, identify edge cases, fix any bugs, test your code and discuss where future enhancements might arise with your interviewer. Ideal time – 5 minutes*

**System Design**

*The interviewer will probe for the ability to build a solid design for a substantial system; quickly identify key tradeoffs in the design; and balance operational performance with customer needs.*

*You want to demonstrate that you can deliver a design in production with considerations of deployments, scaling, failures, availability, and performance.*

*1. That you can build a solid Object Oriented design for a substantial system*

*2. Can be trusted to work independently to build a system for medium size, complex projects*

*3. talking generally about distributed systems, Service Oriented Architectures and n-tiered architectures*

*4. Demonstrates significant breadth/depth of knowledge in a technical area*

*5. Domain Expertise*

*\*Pro Tip – Review your resume and detail all the systems you have delivered throughout your career*

*\*Review the 6 Pillars of the Well Architected Framework – This would help with understanding how systems are designed at AWS.*

[*https://aws.amazon.com/architecture/well-architected/?wa-lens-whitepapers.sort-by=item.additionalFields.sortDate&wa-lens-whitepapers.sort-order=desc&wa-guidance-whitepapers.sort-by=item.additionalFields.sortDate&wa-guidance-whitepapers.sort-order=desc*](https://aws.amazon.com/architecture/well-architected/?wa-lens-whitepapers.sort-by=item.additionalFields.sortDate&wa-lens-whitepapers.sort-order=desc&wa-guidance-whitepapers.sort-by=item.additionalFields.sortDate&wa-guidance-whitepapers.sort-order=desc)

*Here are some system design resources to help you prepare:*

* • System Design Sample Questions and Overview: <https://www.interviewhelp.io/blog/posts/amazon-system-design-interview-questions/>

•System Design Primer: <https://github.com/donnemartin/system-design-primer>

• System Design: <https://www.youtube.com/channel/UC9vLsnF6QPYuH51njmIooCQ>

• Grokking the System Design Interview: <https://www.educative.io>

• Grokking the Object Oriented Design Interview: <https://github.com/tssovi/grokking-the-object-oriented-design-interview/blob/master/object-oriented-design-case-studies/design-chess.md>

**Coding Problem Solving**

*You will receive a high-level technical problem deigned to assess your ability for thinking through problems, identifying alternatives, and weighing different options; seeing potential roadblocks and challenges with solutions. You want to demonstrate that you can:*

* Thinking through the problem
* *Identifying alternatives, weighing and choosing*
* *The ability to identify and recommend trade-offs*
* Seeing potential roadblocks/challenges with chosen solution
* Can decompose a problem into understandable points and Deal with Ambiguity
* Able to describe your reasoning behind their design

**Coding Logical & Maintainable**

*You will be given a technical problem designed to see if you write code that is syntactically correct and does what it is intended to do; builds maintainable and readable code. You want to demonstrate that you can:*

* *Write Scalable & Maintainable Code that is easy to read and understand*
* Syntax correctness, error handling and testing
* Ease of implementation
* Maintainability and readability
* The code should work

**Coding Data Structures & Algorithms**

*You will be given a technical problem designed to see if you can apply various data structures & algorithms; makes good trade-offs; and can combine data structures to solve complex use cases (that scale with level). You want to demonstrate that you can:*

* Understand underlying implementation
* *Understand different data structures and algorithms (and can they make good trade-offs)*
* Asymptotic runtime complexity
* *Basic OO design concepts - not system design*

*Also consider the following:*

* *DS & Algo problems that come up regularly include: graphs, balanced trees, stacks, linked lists, hash tables, depth first / breath first search, traversals, divide and conquer*
* The question is designed to see how well you understand and apply data structures?
* *Consider runtimes for common operations and understand how they use memory.*