**2) Structure of Interview**

Consist of 4 separate 1:1 60-minute interviews with scheduled breaks in between. There may be Amazon employees shadowing for their own development.

-In each round You will be asked 1 Technical Question and 2 Behavioral Questions aligned to Amazon’s LPs

-3 rounds will be dedicated to coding and 1 will be dedicated to systems design. I am unsure of the particular order of your interview so please be prepared.

-30 mins to solve the technical question 15-20 for behavioral questions and 5-10 minutes for you to ask questions at the end.

**3) Technical Best Practices and Recommended Actions**

Your #1 goal is to identify the most optimal solution for coding and systems design and execute the solution correctly via code or design.

1. **Ask clarifying questions to disambiguate the problem**: before starting to write code or design your solution, make sure you fully understand what the interviewer is asking you to do. It’s highly encouraged to ask clarifying questions to fully scope out the problem.
2. **State your understanding and assumptions about the problem**: this can help prevent you from going down the wrong problem-solving path due to a misunderstanding.
3. **Consider alternative approaches and their tradeoffs**: while some problems may have a clear ‘correct answer’, others may have multiple answers that are technically correct. So, the first solution you think of isn’t necessarily the most optimal. Consider different approaches to solve the problem, and state the tradeoffs between these approaches before selecting one.
4. **Consider edge cases**: does your solution apply to different scenarios/inputs, or does it only work under certain conditions?
5. **Consider the scope and scale of your solution:** if the problem statement were to be expanded or changed, could your solution handle the extensions?
6. **Think out loud:** this enables your interviewer to follow your thought process and decision-making.

**4A) Coding Competencies**

1. **Coding:** [**Problem Solving**](https://w.amazon.com/bin/view/InterviewerBarUnification/SDE/CodingProblemSolving/) evaluates your ability to disambiguate a problem and logically break the problem down before arriving at an optimal solution.
   * If you feel like you don’t have enough information to solve the problem, you probably don’t – make sure to ask clarifying questions.
   * The solution you come to is important, but so is HOW you come to that solution. This is where thinking out loud and describing tradeoffs becomes extra important.
   * Consider whether there is more than one way to solve this problem, or if parts of the problem are more complex than you originally thought.
2. **Coding: Data Structures & Algorithms** assesses to your ability to identify and implement data structures and algorithms to optimally solve a problem. Things to consider:
   * Brush up on all common data structures, especially ones you don’t use regularly.
   * Consider the different possible data structures could you use to solve this problem. Is one more optimal than the rest? Make sure you can justify your reasoning!
   * Provide the [runtime complexity](https://www.sciencedirect.com/topics/computer-science/runtime-complexity) and [space complexity](https://www.baeldung.com/cs/space-complexity) of your solution.
3. **Coding:** [**Logical & Maintainable**](https://w.amazon.com/bin/view/InterviewerBarUnification/SDE/CodingLogicalMaintainable/)tests your ability to write clean, workable, efficient, and extensible code.
   * Write correct code and consider whether there is a simpler way to accomplish what you’re trying to do.
   * If someone else came along and looked at your code with minimal context into the problem, would they understand what your solution is doing?
   * When writing your solution, consider what would happen if new requirements or additional complexity were added. Could you easily incorporate these new elements, or would you have to rewrite a significant portion of the solution?

**4B) Coding Best Practices**

The expected output is nothing but the optimal solution. How can your code stop bad inputs from going through?

1. Avoid abbreviating the variables - Use proper variable names that can be recognized. Abbreviations are used for only certain things (http)
2. Avoid unnecessary loop constructs - Keep it simple
3. Always keep your variables closer to their usage - Avoid declaring global variables
4. Be specific while defining a function - referring to what is the main objective of the function
   1. This is referring to be very specific in what you want to convey and what tasks are tracked. So, anyone looking at the code should be able to clearly understand the purpose and the context of the purpose is clearly stated.
5. Don't repeat yourself
6. EDGE Cases
7. Object oriented design

**5) Systems Design**

**Tooling**

1. [**https://miro.com/**](https://miro.com/)

**External Prep Resources**

1. [**https://systemdesignprimer.com/netflix-system-design/**](https://systemdesignprimer.com/netflix-system-design/)
2. [**https://github.com/donnemartin/system-design-primer**](https://github.com/donnemartin/system-design-primer)

**System Design: Scalability & Operational Performance** gauges your ability to gather requirements from a customer and translate those requirements into a fully-functioning, high-level design of the system.

I recommend treating your interviewer as the customer. They will provide an intentionally ambiguous product ask/problem statement because, much like what happens in real-life software development, they are looking for you to help them scope out the problem by asking relevant clarifying questions to gather requirements.

Your goal is to build a SCALABLE system design and identify key metrics to monitor the performance of your system that address your customer’s requirements.

You will be responsible for designing all 3 layers of your system:

API layer,

middle layer for messaging

data access layer + database on the backend.

**Step 1: Gather Requirements**

1. Functional Requirements - Exactly what is required and identify key services
   1. Data storage
   2. What kind of data are we talking about?
   3. How complex computations are we talking about (data manipulation)
2. Non - functional requirements - key attributes
   1. Scalability
   2. Availability (doesn't matter what the load is, but when a user requests info it should be available at any time)
   3. Durability (working with millions of customers data, store their data and secure their data)
   4. High Performance (everything is simple and fast)
   5. No single Point of Failure (server 1 is getting request from user, fetches from database and sends back to user, what if the server goes down) this is unacceptable, multiple instances stored in different servers
3. Trade-offs (consistency versus performance)

**Step 2: Key Performance Indicators**

Identify 2-3 KPI’s you would like to measure to assess the health of your system and how you are going to monitor them

* Performance
* Availability (look for issues using error logs)
* Scalability (add instances and scale back in drop of traffic, automatic scaling)
* Security (this is a big topic and need to address this and be concise)

**Step 3: Build**

In this phase, the candidates should have clarified everything with the interviewer and has good understanding of what is expected out of this system.

**SDE2 Bar for Systems Design**

Looking for API design (articulate API's and what they look like)

Choice of database

Choice of Storage, style of querying

Load balancing (talk about how requests will be routed among various hosts)

Caching (talk about techniques to cache data with higher demand for faster availability while keeping a sensible cap)

Sharding

Understanding when to use nosql and sql

Interaction with data store: how to avoid putting too much load on the data store

**CONCLUSION**

Build a system providing a solution to the problem statement and addresses the following:

* Highly scalable
* No single point of failure
* Briefly talk about key metrics (health of the system) and how they can be monitored

Additional Technical Prep Resources

* <https://www.pramp.com/#/>
* Amazon SIP Workshop <https://meetandengage.com/amazon#event-akv80fazn>
* <https://github.com/donnemartin/system-design-primer>
* <https://systemdesignprimer.com/netflix-system-design/>

**6) Leadership Principles**

Your answers to the Leadership Principles are equally important and equally weighted in your interview. We believe past performance is an indicator of future success and your interviewers want to understand how you are as a leader and how you are able to make decisions with your customers in mind. If you haven’t already I recommend you have 1-2 strong examples aligned to each LP refined to the STAR format that you are audible ready with.

<https://amazonsdeinterviewprep.splashthat.com/>

<https://amazon.jobs/en/software-development-interview-prep?INTCMPID=OAAJAZ100026B#/>

You want your examples to be large in scope, scale, impact and complexity.