Interview Process

- Tech : Coding (Software Engineering) : Depth in specialization.
- Tech : Coding (Software Engineering) : DS, Algo and Problem Solving.
- Tech : Design and Architecture : New Problem
- Leadership : Citizenship, Operational Excellence and Collaboration.

How to Prepare for Coding Interview (Depth in Specialization) :

- Understanding the problem and asking clarifying questions before committing to a solution is the best way to start.
- Providing workable solutions without compromising on code quality, code coverage and covering edge cases adds merit.
- Brushing up on the below topics from existing projects and aligning to real-time problems solved at Uber will help you ace the interview.
    - ❖ Efficiency (Time and Space Complexity)
    - ❖ Simplicity (not complicating the solution, keeping things simple!)
    - ❖ Quality of Abstraction
    - ❖ Production ready, reusable and testable code.
    - ❖ Example questions (1 complex question with e2e coverage, Design a message broker)
    - ❖ Team agnostic
    - ❖ Design a rate limiter (rate limiting, circuit breaker)
    - ❖ Rate Limiter : In a multi threaded system / concurrent user requests world, How do you expose the different facets of rate limits at sub second intervals with scale, in a high scale scenario how do you maintain counters (choice of DS)?

How to Prepare for Coding Interview (DS, Algo and Problem Solving).

- Understanding the problem, asking clarifying questions before committing to a solution and choosing the right Data Structure and algorithm plays a vital role in problem solving; e.g., choosing DFS vs BFS vs Dijkstra's vs Bellman-Ford algorithm and so on.
- Backing the decision of choosing the DS and Algo with valid data points or prior expertise will come in handy.
- Ask clarifying questions and it’s fine to backtrack by choosing an alternate algorithm, it’s ideal to make such changes faster and course correct.
- Our interviewers are there to help you with hints and suggestions, and be open to accommodate the recommendations that’ll help you analyze time & space complexity to arrive at workable solutions.

How to Prepare for Design and Architecture : New Problem

- At Uber, we build systems that solve problems in real time; the underlying architecture across search, incentives, rating, delivery, payments and so on are built on data in motion (batch and streams).It's imperative to understand and appreciate the scale we operate at before arriving at the choice of tools and tech stack.
- Asking relevant questions at the start to understand the scope, breaking the problem into small components that you'd like to tackle, clarifying on the assumptions about peripheral/supporting systems to solve for the most common case first before diving deep into edge cases is one of the best approaches.
- Refreshing your knowledge on Application Layer, Data Layer, API's & Data Models, Volume and variety of data sets, storage systems / store choices, Message Queues / Brokers, Caching mechanism, ACID vs CAP vs BASE, Data Sharding / Partitioning, Security, Product / Platform Trade-Offs and Failure cases are few key areas.
- Opting to take an iterative approach towards scaling up and scaling down, having a logical flow towards solving the problem, incorporating ad-hoc changes / recommendations and backing your design choices by your prior experience will help you ace the interviews.

How to prepare HM Round

Prepare examples on these areas-
Problem-Solving Skills
Collaboration and Leadership
Learning and Adaptation
Collaboration Across Teams
Judgment and Decision-Making
Risk management

You can use these sites to practice your coding and Design.

Some helpful links to review for Design & Architecture :
• http://blog.gainlo.co/index.php/category/system-design-interview-questions/
• https://www.interviewbit.com/courses/system-design/topics/interview-questions/

For all coding rounds you can utilize the following tool to help prepare: www.leetcode.com
One technique to write a better solution is to consider what you would catch if you were code reviewing your code. Consider areas as in this blog post on code reviews.
https://www.interviewbit.com/courses/programming/ - Coding Practice Platform
https://www.interviewbit.com/courses/system-design/ - Design Practice Platform
https://www.geeksforgeeks.org/how-to-crack-system-design-round-in-interviews/ - How to crack System Design Interviews.
https://github.com/shashank88/system_design - System Design preparation Cheat Sheet

https://gist.github.com/vasanthk/485d1c25737e8e72759f - System Design Cheat Sheet
https://hackernoon.com/top-10-system-design-interview-questions-for-software-engineers-8561290f0444 - Sample System Design interview Questions

https://www.careercup.com/page
https://www.programcreek.com/2012/11/top-10-algorithms-for-coding-interview/
https://www.geeksforgeeks.org/
https://www.pathrise.com/companies/uber - You can check and explore this as well..

https://eng.uber.com/training-engineering-interview/ - Navigating the Engineering Interview Process at Uber & Beyond.
https://workat.tech/resources

YouTube Videos Links For Coding & Design : Pls check below and you can also search for more for Coding & Design.
https://www.youtube.com/channel/UCRPMAqdtSgd0Ipeef7iFsKw/featured
https://www.youtube.com/user/tusharroy2525/featured
https://www.youtube.com/channel/UC-vYrOAmtrx9sBzJAf3x_xw
https://www.youtube.com/akshaymarch7

Design IMP Points:

- Design - Ensure you ask clarifying questions
- Clarify on requirements before jumping into the solution
- Don't make your own assumptions, you can always check with the interviewer.
- Point the areas of focus while designing a solution for a design problem.
- Define Functional Requirements / scope of the problem
- Think from the perspective of a holistic solution..
- Think from user perspective and define APIs and schema.
- Check for Non-functional requirements / scale before jumping into a solution.
- Reasons for the choices you make.
- Think before you jump into solution and provide reasons/justification
- Tradeoff Analysis
- Entities/ Attributes / Entity and Data Modelling
- Articulation on functional requirements, Domain/Data Modeling, API Contracts, High Availability, Consistency choices, Datastore choices, Performance tradeoffs, Scalability, Capacity Modeling
- Class structure, Abstraction, Interactions between classes , Choice of data structures, Time and Space Complexity analysis, Design Patterns, De-duplication.
- Understanding of concurrency primitives