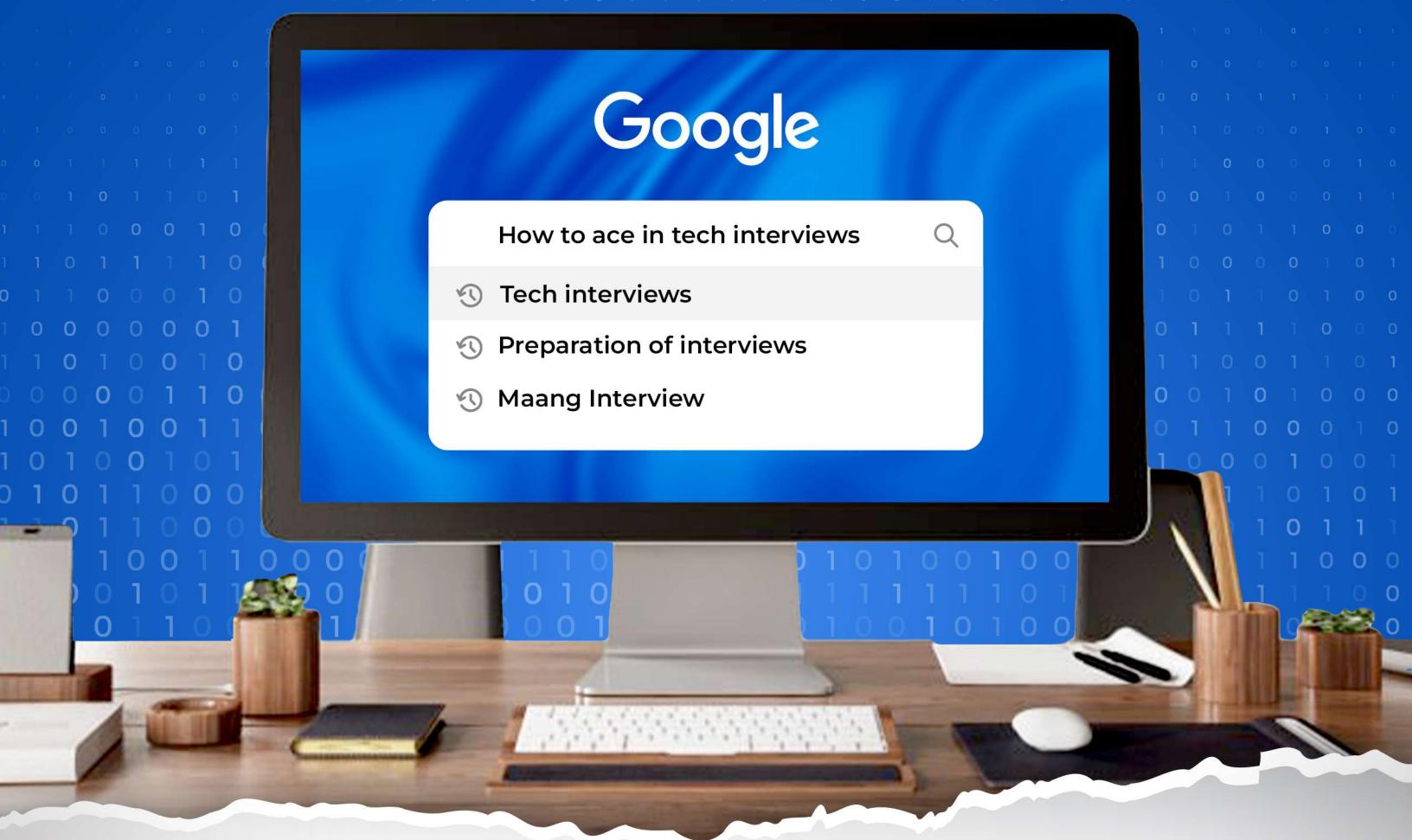




BOSSCODER
ACADEMY



REVANTH
MURIGIPUDI



HOW TO ACE YOUR NEXT TECH INTERVIEW





DISCLAIMER

If you have :

- Covered DSA system design, low-level design, and machine coding
- Developed relevant projects to match job description

Then, this one is for you



PREPARE-REVISE-PRACTICE

- **Fill the gaps** in your knowledge of DSA, System Design, Low-level design, Machine coding

- While polishing, **make flashcards** and **revise** them 10-15 minutes before

- Do company **research** and **thoroughly** read JD → Organization often ask questions :



Why do you want to work with us?

What difference do you find here?

REVISING BEFORE THE DSA ROUND

- Don't grind theory, learn by **practicing problems & solving patterns**

Examples

- Array is sorted?
can you use a binary search? two pointers?
- Running into corner cases with linked lists?
use dummy nodes
- Subarray or substring problem with some conditions? - a sliding window? DP?
- Exploring multiple indices in a matrix at once? - maybe (breadth-first search) BFS?

- Participate in the **contests** & see how well you perform against the time
- Use the leetcode platform efficiently

[Link for → How to use leetcode in an optimized way](#)



DURING DSA INTERVIEW

- **Don't** make assumptions → **ask** questions → **think** of multiple approaches → **make** a decision

Examples

- Prematurely jumping into a **dynamic program approach** when it might be a greedy problem
- Using a **Dijkstra algorithm** when it might be a simple **BFS approach**
- Always begin with a **brute force approach** & gradually move to an **optimal approach**
- Pay attention to any hints given by the **interviewer** & **use smartly**

- Think about the **edge cases** before solving the problem and discuss them with the **interviewer**.
- Tell the **time & space complexity** of all your approaches (brute or optimal)
- Use **proper coding conventions** while writing code following proper **syntax, indentation, naming conventions** etc.



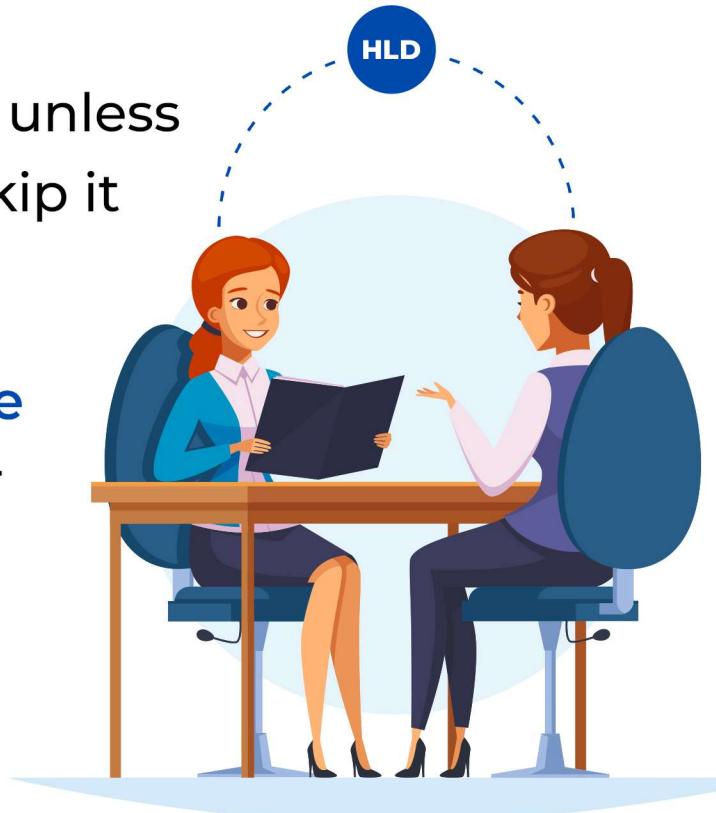
BEFORE THE SYSTEM DESIGN INTERVIEW

- Study **fundamental concepts** like:
 - microservices
 - scaling
 - caching
 - CAP theorem
 - sharding
 - containerization
 - types of databases
 - message queues
 - load balancers, etc
- Watching lots of **mock interviews**, builds intuition of how to **approach** a **design problem** from the ground up



DURING THE SYSTEM DESIGN INTERVIEW

- Drive **functional/non-functional** requirements discussion in the beginning
- Do **capacity estimations** unless the interviewer asks to skip it explicitly
- Consider trade-offs, **while choosing** one entity over another



Examples

- ❑ Quote Logical Explanations for →

Why do you prefer a NoSQL DB
instead of a relational?

Why use a WebSocket instead
of HTTP long polling?



- Start from the **most basic component** &
gradually build on it!

Examples

- Load balancer to route traffic to services, & one service talking to the database, another service pulling messages from a message queue, etc.

- Adapt your **architecture/design** on the fly as per the discussion

- Ask if the interviewer wants you to write a **DB schema** as well?

- Explain the **API contracts**, if you have got time left!

- **System design** interviews are mostly discussion-driven, it's important you **constantly** discuss your thoughts

BEFORE LLD INTERVIEW

- Learn **SOLID principles** and understand design patterns

Examples

Creational - singleton, factory, builder

Structural - adapter, decorator, facade

Behavioral - observer, strategy, command

- Practice class designs for famous interview problems to understand the flow

Examples

Designing movie booking application/car rental system/parking lot system



DURING LLD INTERVIEW

- Clarify the **functional requirements** before designing (some interviewers expect you to write API contracts in **LLD rounds** as well, get it clarified)
- Start with a **very high-level class**, and drill down into components one by one as you seem fit.

Examples

- For a movie booking **LLD problem**, start with a theater class containing a list of shows/screens objects, further containing seats, movie details, etc.

- Talk to the interviewer if they want you to **write pseudo code** for some functionality or not
- Remember, there is no one fixed solution to **design-related problems**, it is entirely discussion & requirements driven!



BEFORE MACHINE CODING INTERVIEW

- Learn to write **proper executable code** for popular problems → snake and ladders, tic tac toe, chess, parking lot, elevator system, in-memory database & search engine, distributed cache, etc
- Take **1-2 hours** to at least write one **functionality** end to end & execute it against multiple test cases
- While practicing, read other's code by searching the problem statement from **Leetcode** or **Github**



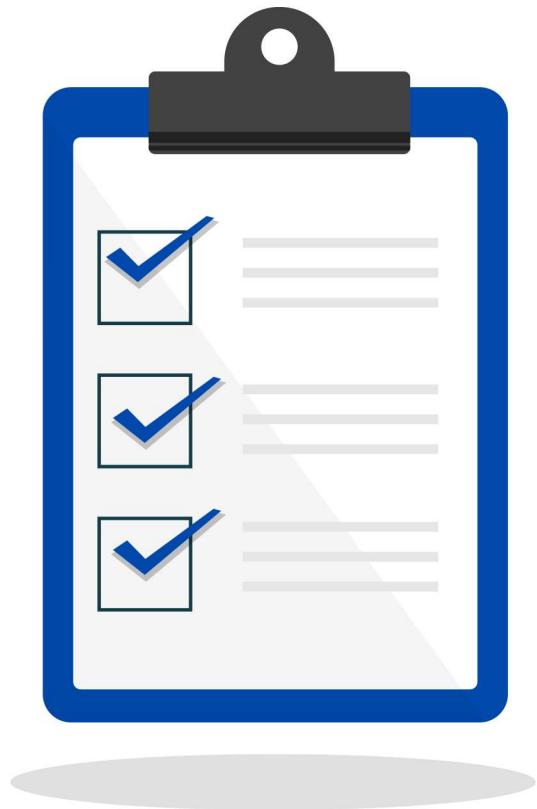
DURING MACHINE CODING INTERVIEW

- **Do not over-commit** as you have only a couple of hours out of which **20-30 mins** go away for discussing requirements themselves
- Start with a barebones structure using design patterns and **SOLID principles** and code your way up to **functionality**
- Aim to finish at least **1-2 core features** from requirements.
- Don't forget to **execute your code** for multiple test cases including the edge cases

MANAGERIAL AND BEHAVIORAL ROUND

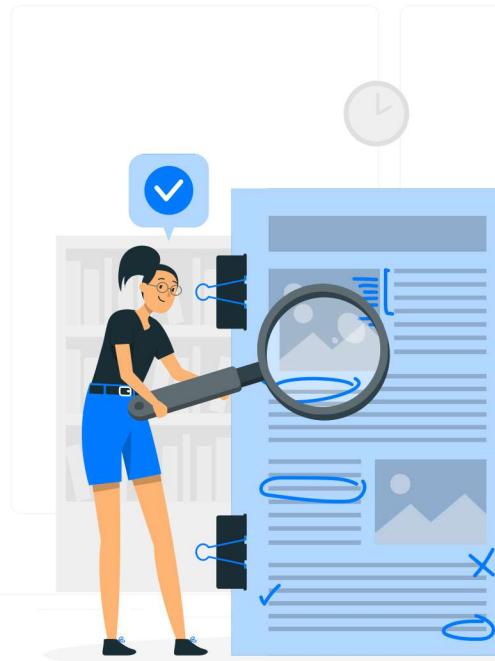
- Use the **STAR approach** and “STAR” stands for “Situation, Task, Action, Result
- Answer by **describing the project (situation)**, explaining what you had to deliver **(task)**, what steps you took to complete **(action)**, and the outcome **(result)**.
- For this Study the **architecture** of your project end-to-end. Brief the **project's relevance** in the real world.

- Here's your list of all imp frequently asked questions
- Prepare a **30 sec** to 1min elevator pitch for avoiding the mess



KNOW YOUR RESUME

- Be thorough with everything you have **mentioned**
- Keep some points handy for
 - “Tell me something **NOT** mentioned in your resume”
- Know the **outcome** and **impact** of the project on business value
- Mismatched information creates **wrong impression**



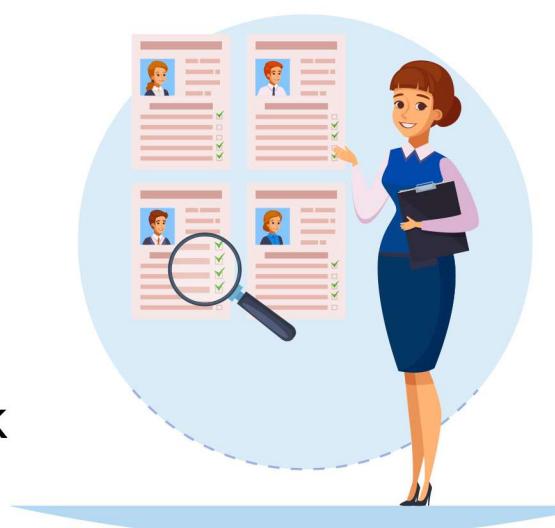
GIVE MOCKS INTERVIEWS

- Take feedback **constructively** & **focus** on mending the mistakes
- Record it, **Watch** + **Listen** to your interview to asses body language tone (em, uh) and speech errors
- **Practice mocks** till you get confident enough to ace any interview



AT THE END

- **Frame** a few questions to ask the interviewer at the end of the discussion
- Here is your [list](#) of questions to be put at the end
- Once you are done, reflect back with improvement
- Asking **relevant questions** shows **your interest** in the role & organization, so **make it count**



MISCELLANEOUS

- Provide **concise responses**. No lengthy stories
- **Summarize answers** in 2-3 mins unless probed for further explanation.
- **Be honest**, If you don't know about some topic, **answer smartly**

“I have not worked on Redis, but yes I know a little bit about cache and I am ready to learn”





HAVING A **RIGHT MENTOR** ON **YOUR SIDE, IS NO LESS THAN A BLESSING**

Bosscoder has got **top product company** mentors, for assisting you **24/7**, throughout your journey in becoming a **great** Software Engineer

- ✓ **Well-structured** curriculum - DS+Algo, CS fundamentals & System Design
- ✓ **Live** Classes
- ✓ **1:1 Mentorship & Mock Interviews** with experts
- ✓ **Career Support:** Resume optimization, Community referrals
- ✓ Industry Relevant **Projects**

Within a span of 7 months, you will develop skills + confidence + hands-on experience to grab your dream job.



WHY **BOSSCODER**

-  **200+ alumni placed at Top product-based companies**
-  **More than 120% hike for every 2 out of 3 working professional**
-  **Avg package of 22 LPA**

I got placed at ByteDance Singapore and I am very thankful to **Bosscoder Academy**

Irshad
 **ByteDance**



Bosscoder gives you everything starting from DSA, LLD and HLD followed CS Fundamentals

Dheeraj Barik