Playlist – Engineering Interview Prep Guide

A Quick Note Before You Dive In

This guide is a general prep resource. It won't cover every single topic or question and please don't feel like you need to memorize responses. We want to see how *you* think, not how well you've rehearsed. Come curious, open-minded, and ready to collaborate.

Can you use AI?

We love AI and encourage its use *once you're here*. But during interviews, we want to see how ***you*** solve problems, unaided. Please **do NOT use AI** or other tools during interviews unless specifically asked by the interviewer. Cameras must stay on, and we monitor off-screen activity. If there's a legitimate reason you need AI assistance, just let your interviewer know and they'll guide you how best to incorporate it.

Our process typically includes five core interviews (*not always in this order*):

Interview: System Design / Decomposition

This interview is about designing a set of data models and API endpoints to facilitate an application. There are two main user types with different use cases you will cover.

Your tasks:

- Think about availability, scalability, and consistency.
- Start simple—add complexity as new info is introduced.
- Communicate your decisions and trade-offs clearly.

What we're looking for:

- Clear mapping from requirements to data models and endpoints.
- Ability to explain trade-offs and design choices.
- Awareness of concurrency, idempotency, and validation challenges in reservation flows.
- Effective communication and adaptability as scope expands.

💡 *Tip:* Don't try to cover every detail upfront. Start with a minimal design and expand it as new requirements are introduced. Listen to feedback or hints.

Interview: Technical Design Review

This interview focuses on how you design and build code through tests.

Your tasks:

- Work with the interviewer to define requirements in terms of test cases.
- Write tests before writing the implementation.
- Iteratively build a solution by letting failing tests guide your code.
- Refactor

What we're looking for:

- Understanding of the TDD cycle.
- Ability to think in small, incremental steps.
- Balance between writing enough tests and keeping progress moving.
- Clarity in explaining your design and coding choices.

💡 *Tip:* Don't aim for a perfect solution up front. Start simple, get a failing test, make it pass, then build from there. The interviewer is looking for your thought process, not just the final code.

Interview: Code Reading & Refactoring

This exercise is framed as role play: a teammate has asked you to review their code before merging. The code works, but it has problems.

Your tasks:

- Identify issues with readability and structure.
- Suggest improvements around safety, defensive programming, and testability.
- Make targeted refactors to show your thinking.

What we're looking for:

- Ability to spot code issues and explain trade-offs.
- Suggestions that make the code safer, cleaner, and more maintainable.
- Awareness of testing and architecture considerations.
- Clear, collaborative communication

💡 *Tip:* You don't need to rewrite the code from scratch. Focus on identifying problems and explaining *why* your suggestions matter — just like you would in a real review.

Interview: Technical Implementation

You'll work with a piece of code that is almost correct but contains a few bugs.

Your tasks:

- Run the tests and interpret failures.
- Debug and fix logical errors.
- Make a small enhancement.

What we're looking for:

- How you reason through unfamiliar code.
- Ability to debug step by step.
- Use of systematic approaches (stack traces, print statements, walking through input/output).
- Clear communication of your thought process.

💡 *Tip:* Think of it like stepping into a teammate's codebase for the first time. Be methodical, explain what you're doing out loud, and don't worry if you need to test a few hypotheses.

Interview: Non-Technical Conversation

We want to understand how you work, collaborate, and grow. We'll talk about past experiences and what motivates you.

Your Tasks:

- Think of moments where you made an impact or grew
- Be ready to talk about teamwork, challenges, and inclusion
- Be open and authentic about how you like to work

What we're looking for:

- How you collaborate and lead
- Communication and team mindset
- Alignment with our values and mission

Our Core Values:

- **Impact Over Effort:** Focus on outcomes, not hours
- **Bias for Action:** Learn by doing
- **Inclusion:** Support diverse voices
- **Collaboration:** Discuss, align, move forward

- **Resilience:** Stay grounded through change

Final Tips

General Advice for All Coding Interviews:

- Think out loud. Share what you're testing and why — communication matters as much as code.
- Use examples. Walk through input/output scenarios to verify your reasoning.
- Ask clarifying questions. Requirements can be clarified just like in real work.
- Stay calm when stuck. Break down the problem and try one small step at a time.
- Collaborate. Treat the interviewer like a teammate.

How to Prepare:

- Practice debugging by intentionally breaking small programs and fixing them.
- Review open-source code and think about what feedback you'd give and how.
- Brush up on SQL best practices, defensive programming, and basic refactoring strategies.
- Sketch a simple API design with models and endpoints in 20 minutes (time constraints can be hard!)

Final Note:

We want you to succeed and enjoy the process. If you get stuck, your interviewer will provide hints and work with you to come to a solution - it's a collaborative session, not a trap!