Coding and Design Tips

**Coding:**

* There will be three coding questions asked – focusing on problem solving, logical and maintainable code, and data structures and algorithms.
* The onsite session will require coding by hand -  you will not be able to use any sort of IDE or Compiler to test your code. I would like to emphasize any practice that may be needed when coding by hand – like brushing up through a code test like [Leet Code](https://leetcode.com/) or [Codefights](https://codefights.com/)  or [InterviewBit](https://www.interviewbit.com/). You will want to make sure that your code is not pseudo.
* The team is open to any object oriented-language – we want to ensure you are writing in whichever language you are most comfortable with –  please review Syntax so that it is as syntactically correct as possible although small errors here and there are not a big deal!
* Review DS/Algorithms – Please have an understanding of when and where to use certain DS and Algorithms. Know time and space complexity tradeoffs.
* Discuss your thoughts out loud – interviewers are looking to see the depth and breadth of your knowledge. They can’t assess what you aren’t telling them, and it doesn’t give an opportunity to redirect candidates or provide tips if they don’t know their train of thought so please feel comfortable walking through your thoughts out loud!
* Ask Clarifying questions– many of the questions you are asked may be intentionally vague and require clarity before solving. Asking questions is encouraged!
* Test your code (if applicable) –candidates should do their best to test the logic of their code when possible. A simple unit test on the white board will go a long way as it will require the code author to think through inputs, outputs, and error conditions
* Think through – is your code extensible and maintainable?

**Design:**

* When thinking through a design problem, remember it has 3 components below. Also be aware that we will want to talk about an existing System Design as well as a new system design for a total of 2 design problems.

1. **Define Requirements and Goals of the System –** here is where you will want to askclarifying questions and with requirement gathering – remember to think through what the system will do - Narrowing the scope: what the system won’t do - Get a complete picture of the service – depth & breadth
2. **Make design considerations -** What is critical from a computing standpoint? Identify major components and sub-components. Anticipate scaling as early as possible. Upsides VS. Downsides
3. **Constraints and bottlenecks** - How can this problem expand – where does it scale to?

* **Traffic:** “1M users increases to 100M, what does that mean for TPS”
* **Latency:** Time from browser to server, caching, pre-fetch
* **Memory:** Vertical Scaling, limited resources, downtime, upper limits
* **Security:**  Privacy, Abuse, etc.
* **Storage:** “How many objects will we store per year, and how big?
* **Databas**es: Horizontal scaling**,** How does data behave across a distributed network?

**General advice for System Design:**

* Be a Consultant – imagine your interviewer is a customer you’ll be building something for, consider a tangible system, from requirements to post-production, and what that means.
* Drive the conversation by asking great questions and validating assumptions.
* Know Why – Defending design tradeoffs is critical.
* Interesting Discussion – show advanced or unique edge cases – validate your assumptions and ask the right questions
* Consider the customer – is this a good experience? What purpose does it serve?