OpenAI Software Engineer Interview Preparation

Note: You are free to use the language of your choice and to Google standard library functionality if needed, but we don't allow the use of AI tools during the interviews.

Generic Coding (CoderPad) General Guidance:

- Test your preferred programming language in CoderPad before the interview to familiarize with setup (ie importing libraries, etc.) to optimize interview time
- Data structures and algorithms - be familiar with how they work, optimal use cases, and how to implement them in code:
- Debugging & testing - debuggers, logs, instrumentation to identify/fix bugs, unit tests, integration tests, test coverage
- Familiarize yourself with the test-driven development approach where you write tests before implementing features. This will help ensure your solutions are robust and cover all edge cases.
- Practice managing your time effectively during mock interviews to ensure you can address all parts of a complex question. Provide a working solution first and then refine it if time allows.
- Strive for clarity and simplicity in your code. Use appropriate naming conventions, keep functions small and focused, and avoid unnecessary complexity in your solutions.
- Be ready to discuss the theoretical aspects of your solution, such as time and space complexity, or the trade-offs between different data structures you might use.

What to Expect:

- Questions centered around data structures (arrays, lists, trees, maps, sets, stacks, queues, graphs) and their tradeoffs.
- Algorithmic thinking involving sorting, searching, traversal, etc.
- Code should be clean, readable, and well-structured.
- Questions where you design custom data structures optimized for specific read/write patterns.
- You may be asked to track or manipulate time-based data or serialization-related data.
- Understand how to structure and access data efficiently.
- Review serialization concepts (e.g., JSON, Protobuf), and how to serialize/deserialize in your language.
- You may be given existing code to analyze and improve.
- Debugging scenarios, especially involving concurrency and distributed systems.
- Design-focused coding problems, especially in distributed system contexts.

- Problems that emphasize high-scale scenarios, concurrency, or networking concepts (e.g., CIDR, IP addressing).
- Some interviews involve non-executable code, focusing on logic and design rather than syntax.
- Others involve toy scenarios like a made-up programming language or custom type systems.

How to Prepare:

- Review key libraries and built-in functions in your preferred language (e.g., Python's collections, Java's HashMap, etc.).
- Practice reading unfamiliar code and identifying bugs or design flaws.
- Review debugging tools and techniques in your language of choice.
- Think through common concurrency issues: race conditions, deadlocks, and shared state.
- Refresh networking basics (IPv4, CIDR blocks, subnetting).
- Think through common backend problems like data consistency, sharding, replication, etc.
- Practice designing code structures without relying on running them.
- Be familiar with generics and type matching (e.g., templates in C++/Java).
- Be ready to discuss your approach clearly, explaining decisions and handling edge cases.

System Design (Excalidraw) General Guidance:

- Consider practical, real-world scenarios and gather requirements before diving into your system design. Understand the nuances and potential challenges that may arise in a live environment.
- Be prepared to discuss the trade-offs involved in your design choices. Showcase your understanding of the balances needed in system design, such as between consistency and availability.
- Clearly articulate your thought process during the design discussions. Be ready to defend your design decisions logically and convincingly.
- Develop and demonstrate a holistic understanding of the system, ensuring that each component, from databases to APIs, is designed with consideration of the entire system's functionality and performance.
- Concentrate on designing systems that can gracefully handle growth and increased load, ensuring long-term sustainability and performance. Consider potential risks and failure scenarios, asking questions like "What happens if component X fails?" to enhance the system's resilience.

What to Expect:

- Problems that begin with high-level goals like availability, durability, or latency.
- Strong emphasis on requirement analysis and clarification.
- You may be expected to design production-grade infrastructure using open-source tools or cloud provider services (AWS, GCP, Azure).
- Interviews often focus on one or more core components: Kubernetes, networking, queues, databases, etc.
- These questions focus on scaling, fault tolerance, and tradeoffs in distributed systems.
- No single right answer—interviewers care more about your thinking and justifications.
- These problems involve designing or adapting a system to run more efficiently, often across multiple machines.
- Interviewers may continually shift the constraints to see how you react.
- You may be expected to critique your own solution and iterate.

How to Prepare:

- Practice probing for requirements: what are the exact SLAs, traffic volumes, consistency needs, failure tolerances?
- Learn to break down ambiguous problems into constraints and tradeoffs.
- Use frameworks like PACELC, CAP theorem, or the "4 Pillars": availability, scalability, consistency, and maintainability to structure your thinking.
- Gain deep familiarity with at least one infrastructure layer (e.g., how Kubernetes works under the hood, how queues handle retries and DLQs).
- Understand cloud-native design patterns like service meshes, autoscaling, observability, etc.
- Be able to reason about infrastructure as code, deployments, and CI/CD pipelines.
- Study classic topics: leader election, consensus protocols, eventual consistency, data partitioning.
- Practice drawing system diagrams and explaining each component's role.
- Understand common failure modes: network partitions, node crashes, replication lag, and how to mitigate them.
- Practice iterative design: start simple, then refine for performance, scale, and resilience.
- Stay flexible and avoid anchoring on a single design.
- Be ready to analyze bottlenecks—where does your design break down as load increases?

OpenAI Values

These values define what we consider to be the most important things. They guide our decision-making. We believe that channeling these values is the most promising way to achieve our mission.

- Humanity first: Working at OpenAI means being part of a team that is passionate about benefitting people and society through our work. We build AI to elevate humanity.
- Act with humility: Humility reminds us to recognize the limits of our own knowledge and to remain open to new ideas, perspectives, and the possibility of being wrong. This mindset influences our iterative approach to deployment, and the reintegration of feedback into our research.
- Feel the AGI: AGI will be powerful in an unprecedented way, with potential for upside and downside. Building it requires rigor and discipline, boundless imagination, and a deep sense of responsibility. We approach research and product development with respect for the transformative technology we are helping create, and the coupled benefits, changes, and risks that come along with it.
- Ship joy: Through our research we develop products that can transform how people live their lives. Our technology reflects an internal culture of optimism for the future and stewardship of our mission.

OpenAI Operating Principles

These principles define how we work together. They are important for establishing our operating culture.

- Find a way: We believe in finding a way to do the things that matter. We give agency to individuals and teams to find an approach that works. Ideas come from anywhere, regardless of title or tenure.
- Creativity over control: We look to find creative solutions, even if sometimes imperfect, over rigidity and control. Our approach to solving problems is based on best principles (first principles and best practices).
- Update quickly: We come in with a hypothesis and are willing to change our approach as we receive new information. We seek truth and adapt knowing that flexibility is key to progress on the path to AGI.
- Intense focus: We are hard workers who are here to make an impact on the world. This intensity and resilience are pivotal to the mission, while clarity and focus enable us to make hard decisions.