Frontend Engineer Technical Competency Prep Guide

Coding - Frontend Technologies

- Creates an accessible solution, or speaks in depth about strategies that would render an accessible UI experience
- Demonstrates web-focused Frontend fundamentals: JavaScript, HTML, CSS
- Demonstrates an ability to create a solution that will address a wide variety of customer needs and situations (e.g. device and browser support, internationalization/localization, bandwidth concerns)
- Leverages industry standards to create a solution that is easy to understand, maintain or improves customer experience
- Implements a solution that is optimized for scale or reuse (e.g. does not violate system constraints, such as reusing ID attributes in HTML document; number of repaints scales linearly with respect to data or network traffic; network traffic scales linearly with regards to user interaction frequency)
- Identifies potential shortcomings and discusses trade-offs in the approach and technologies used
- Describes testing practices that are practical and able to catch issues which may be found in any part of the solution

Example Question: Tic-Tac-Toe Game Engine
Implement a game engine to determine the winner of a tic-tac-toe game.

Rules

- 2 players take turns placing either an X or an O in one of the squares on the grid
- If either player gets 3 of their symbols in a row, they win the game:
    - o You should have logic for validating a user's move to ensure it's legal (can't place a symbol over another symbol) o Solution should easily be modified to handle an NxN Grid
    - o Should handle edge case validation (clicking the same spot twice)
    - o Should be able to talk about a couple different approaches to solving the grid
    - o Minimal help provided

Expectations

- Ex. "Return the number of valid moves for a robot 2D game board, given some simple rules" (question ambiguously asking if you are familiar with a 2D array)
- Ex. "Write a program that takes 2 numbers as arrays: [1, 2, 3] and [3, 4, 5] and returns the sum: [4, 6, 8]"

Tips:

- Be very familiar with best practices
- Abstracting helping methods to help extend your code

Coding – Frontend Data Structure and Algorithms

- Uses optimal data structures and algorithms to solve the problem
- Identifies potential shortcomings and discusses tradeoffs with different data structures and algorithms
- Justifies why the selected data structures and algorithm was used

- Demonstrates solid grasp of runtime and space complexity tradeoffs even if not perfectly accurate O(n) syntax
- Understand common algorithms: traversals, Hash Maps, Lists, Trees, Priority Queues, divide and conquer, when to use breadth first vs. depth first, recursion, etc.
- Discuss runtimes, theoretical limitations and basic implementation strategies for different classes of algorithms

Example Question:

- "Maintain a list of recently visited web pages by a user, in descending order of time, without duplicates"?
- "Write code for a question to print the left view of a binary tree"?

Coding – Frontend Logical and Maintainable

- Creates simple code (e.g., leverages reuse, properly formatted, no improper coding constructs
- Creates maintainable code (e.g., quickly able to trace impact of changes, clear variable naming conventions)
- Readability - allows for code to be well structured, field names are clear/understandable, easy to follow the logical relationships.
- Complexity of methods/testability - allows for code Elements have clear single responsibility and usage of right assertions.
- Design Patterns - Question allows candidate to be able to speak to multiple different patterns and pros/ cons of one over another. Can articulate reasoning for choice of specific pattern. Code is elegant, logical, and follows solid design principles. Can break down hierarchies into named functions
- Extensibility - code allows for candidate to be able to articulate how it can be extended in the future beyond specified requirements.

Example Question*:* This virtual marketplace has products that are stored hierarchically within their respective categories.

- Products have various metadata, like current price or if it's prime-eligible.
- A dress might be located under Clothing > Women > Dresses, with a price of 20 and it is prime-eligible.
- It should be easy to add additional criteria to search by in the future.
- But initially only needs to handle querying by category or max price***.***

Tips: It is expected you ask clarifications about how the products are stored to be searched, since that may not be provided.

Frontend System Design (Scalability and Operational Performance)

- Asks clarifying questions to scope-down and define requirements
- Creates a UI/UX design for a system that fulfills captured requirements (e.g., constraints, scalability, maintainability, performance, security, accessibility)

- Scalability – through requirement gathering clarify how this design should scale i.e. geographical coverage (regional/multi-regional hosting), how many expected users or traffic, etc.
- Designs for operational performance and plans for failure and can measure the results (e.g., metrics)
- Identifies potential shortcomings and tradeoffs with different designs (e.g., performance, fault, tolerance, device limitations, dependencies)
- Able to showcase interaction across Frontend and Middle-Tier layers.
- Able to articulate Frontend/Backend contraction – speaking on API Pagination.
    - Speak on data caching and API Pagination, Nodes (how many required for scaling)
- Able to assess scalability and performance of a frontend design when data results go from millions to billions.

Example Question: Design the UI for a virtual storefront?

Scale recommended results from 1,000 to 1,000,000.

Resource: FE System Design Video Tutorial