Technical Interview Questions

Technical Interview Questions for Fokuslah Developer

Category: System Architecture & Design Patterns

Question 1: Feature-Sliced Architecture

Our codebase is organized using a "feature-sliced" architecture (e.g., `/features/lessons`, `/features/profile`).

- a) Can you explain the primary benefits of this approach compared to a more traditional layer-based structure (e.g., a single `/components`, `/api`, `/hooks` folder for the entire application)?
- b) What are some potential drawbacks of the feature-sliced pattern, and how would you mitigate them as the team and the number of features grow?

Question 2: API Design Philosophy (RPC vs. REST)

We use Hono to create an RPC-style API client for the frontend. This gives us end-to-end type safety.

- a) What are the trade-offs of this RPC approach versus designing a conventional RESTful API for this application?
- b) In what scenarios would a REST or GraphQL API have been a better choice for "Fokuslah"?

Question 3: Idempotency in API Design

The lesson submission endpoint is designed to be idempotent using an `attemptId`.

- a) Why is idempotency crucial for this specific `submit` endpoint?
- b) Can you walk me through how you would implement this idempotency check at the database level? What are the key data points you would need to store and verify?

Question 4: Refactoring a Critical Function

The `submit` endpoint was refactored to use an "early return" or "guard clause" pattern. The original code had more nested if/else statements.

- a) From a code quality and maintenance perspective, why is the early return pattern often superior for complex functions?
- b) How does this refactoring also help with static analysis and type inference, as we saw with TypeScript?

Category: Frontend & State Management

Question 5: Data Flow and Caching with React Query

Describe the entire data flow, from the user's perspective, when they correctly answer a question on the `/lessons/[id]/[questionId]` page.

- a) How do `useMutation` and `useQueryClient` (specifically query invalidation) work together to ensure the UI (e.g., streak count, XP, lesson progress) reflects the new state?
- b) What potential race conditions or caching issues could occur in this flow, and how would you prevent them?

Question 6: Extensible Component Design

Imagine our `ProblemSolver` component, which currently handles multiple-choice and text-input questions. We want to add a new "drag-and-drop ordering" question type.

- a) How would you architect the `ProblemSolver` component and its related components/types to be easily extensible for this new question type without requiring a major refactor?
- b) Describe the data structure you would use for this new question type in the `Problem` model in `schema.prisma`.

Category: Database & Data Modeling

Question 7: Advanced SQL Querying

Our Prisma schema tracks individual XP gains. We want to build a weekly leaderboard.

- a) Can you outline how you would write a single, efficient SQL query to get the top 20 users based on the total XP they earned in the last 7 days?
- b) What database indexes would be critical for making this leaderboard query performant as the number of users and XP events grows into the millions?

Question 8: Ensuring Data Integrity with Transactions

The process of submitting an answer involves multiple, related database writes: creating a `UserProblemAttempt`, updating `UserProfile` (total XP), and updating `UserStreak`.

- a) Why is it risky to perform these as separate, sequential database calls?
- b) How would you use Prisma's database transaction features (e.g., `$transaction`) to group these operations, ensuring that they either all succeed or all fail together, thus maintaining data integrity?

Category: Technical Implementation & Problem Solving

Question 9: Debugging a Type Inference Issue

We encountered a persistent TypeScript error where `InferResponseType` from our RPC client failed to detect changes in the backend route's return shape.

- a) What are the common reasons why a type inference system might fail, particularly with complex functions that have multiple conditional branches?
- b) If you were tasked with debugging this, what would be your step-by-step process to identify the root cause?

Question 10: Streak Calculation Logic

Our streak calculation logic needs to be robust. It handles multiple cases: starting a streak, continuing it, breaking it, and multiple activities on the same day.

- a) Walk me through the edge cases you would consider when writing unit tests for this streak logic. What specific scenarios would you test for?
- b) The logic relies on comparing UTC dates. What potential issues could arise if we didn't normalize all dates to UTC midnight, and how might timezones affect the user experience?

Question 11: Scalability Bottlenecks

If our app suddenly grew to 50,000 concurrent users, what are the top 2-3 technical bottlenecks you would anticipate in our current architecture?

- a) For each bottleneck you identify (e.g., database reads, API response times), propose a specific, actionable solution (e.g., implementing a caching layer with Redis, database read replicas, optimizing specific queries).