Question 1 of 15

Which API protocol should you default to in system design interviews?

1 GraphQL

2 WebSockets

3 gRPC

[✓ 4 REST]

Correct!

REST should be your default choice as it's well-understood, has great tooling, and works for 90% of use cases. Only choose alternatives when you have specific requirements that REST can't meet.

Question 2 of 15

REST resources should represent actions like 'createUser' or 'updateBooking' rather than things in your system.

1 True

[✓ 2 False] ✓

✓ **Correct!**

REST resources should represent *things* in your system (users, bookings, events), not *actions*. Instead of 'createUser', you'd POST to '/users'. Instead of 'updateBooking', you'd PUT to '/bookings/{id}'.

Question 3 of 15

When should you use path parameters vs query parameters in REST APIs?

1 Use path parameters when the value is required to identify the resource ✓

2 Always use path parameters for better performance

3 Use them interchangeably

4 Always use query parameters for flexibility

Correct!

Use path parameters when the value is required to identify the resource (e.g., '/events/{id}/tickets'). Use query parameters for optional filters or modifiers (e.g., '/events?city=NYC&date=2024-01-01').

Question 4 of 15

HTTP PUT requests are idempotent, meaning calling them multiple times produces the same result.

[✓ 1 True]

2 False

Correct!

PUT is idempotent - sending the same data multiple times results in the same final state. This is important for network reliability when clients retry failed requests.

Question 5 of 15

Which pagination approach is more stable when new records are being added?

1 Offset-based pagination

2 Both are equally stable

3 Neither handles new records well

[✓ 4 Cursor-based pagination]

Correct!

Cursor-based pagination uses a pointer to a specific record instead of counting from the beginning, making it unaffected by new records being added during pagination.

Question 6 of 15

When should you consider GraphQL over REST in system design interviews?

1. Always, as GraphQL is more modern
2. [✓ When you see clear over-fetching or under-fetching problems]
3. Never, REST is always sufficient
4. When you need better performance

Correct!

Consider GraphQL when the interviewer mentions scenarios like different data needs for mobile vs web apps, or when you need to avoid over-fetching and under-fetching problems.

Question 7 of 15

gRPC uses JSON for serialization to ensure compatibility with web browsers.

1 True

[✓ 2 False]

Correct!

gRPC uses Protocol Buffers for binary serialization, not JSON. This makes it more efficient but less compatible with browsers, which is why it's typically used for internal service-to-service communication.

What is the primary advantage of Protocol Buffers over JSON?

1 Better browser support

2 Easier debugging

[✓ 3 Smaller size and faster parsing]

4 Better human readability

Correct!

Protocol Buffers use binary encoding that results in smaller message sizes and faster parsing compared to JSON's text-based format, making them ideal for high-performance service communication.

Question 9 of 15

Authentication verifies identity while authorization verifies permissions.

[✓ 1 True]

2 False

Correct!

Authentication proves the user is who they claim to be (identity verification), while authorization checks if that authenticated user is allowed to perform the specific action they're requesting (permission verification).

Question 10 of 15

When should you use API keys vs JWT tokens?

1 Always use API keys for better security

2 Use JWT for user sessions, API keys for service-to-service communication ✓

3 They are interchangeable

4 Use API keys for user sessions, JWT for services ✗

Incorrect.

Use JWT tokens for user sessions in web/mobile applications as they can carry user context and be stateless. Use API keys for internal service communication and external developer access.

Question 11 of 15

JWT tokens require a database lookup to verify user information.

1 True

[✓ 2 False]

Correct!

JWT tokens encode user information directly into the token itself and are signed with a secret key. You can verify authenticity and read user info without database lookups, making them stateless.

Question 12 of 15

What HTTP status code should you return when rate limits are exceeded?

1. 400 Bad Request
2. [✓ 429 Too Many Requests]
3. 500 Server Error
4. 401 Unauthorized

Correct!

429 Too Many Requests is the appropriate status code when a client exceeds rate limits. This clearly indicates the nature of the problem and allows clients to implement appropriate backoff strategies.

Question 13 of 15

In RBAC (Role-Based Access Control), how are permissions typically assigned?

1 To API endpoints directly

[✓ 2 To roles, which are then assigned to users] ✓

3 Directly to individual users

4 Through database queries

✓ **Correct!**

In RBAC, permissions are assigned to roles (like 'customer', 'admin'), and then users are assigned to those roles. This simplifies permission management and ensures consistent access control.

Question 14 of 15

API keys are the best choice for user-facing mobile and web applications.

1 True

[✓ 2 False] ✓

✓ **Correct!**

API keys are almost never the right choice for user-facing applications. Users shouldn't manage long cryptographic strings, and API keys don't expire or carry user context like user sessions need to.

Question 15 of 15

What should you focus on most during the API design section of a system design interview?

1 [✓ Demonstrating solid engineering judgment with reasonable API choices] ✓

2 Using the most cutting-edge protocols

3 Perfect API specification with all edge cases

4 Memorizing every HTTP status code

✓ **Correct!**

API design in interviews is about demonstrating solid engineering judgment, not creating perfect specifications. Focus on choosing the right protocol, modeling resources clearly, and showing understanding of basics like authentication and security.