System Design Interview Framework: ‘Build X’ (mini)

> **Note**: This “mini” framework lists the **core topics** you should cover within a **40–45 min** system design interview. For deeper exploration of any topic, refer to the full framework document.

0. Real-World Framing (1 min)

- Acknowledge interview vs real-world constraints.
- Emphasize pragmatic choices grounded in team capabilities.
- Suggested opening: balance best practices with realism.

1. Scope & Assumptions (5 min)

- Define the business problem, objectives, and stakeholders.
- Identify primary user personas and MVP vs full feature use cases.
- Establish success metrics: technical KPIs and business outcomes.
- Note key constraints: budget, platforms, compliance requirements.

2. High-Level Architecture (3 min)

- Sketch the Four-Tier Model: *Client* → *Edge* → *Business* → *Data*.
- Summarize the request flow through these tiers.
- Highlight the purpose of each tier.

3. Frontend (3 min)

- Platform choices: Web, iOS/Android, cross-platform.
- Data fetching strategies: REST vs GraphQL vs gRPC.
- Client-side caching and offline support essentials.
- Session management and notification basics.

4. Edge / Gateway Layer (3 min)

- Ingress responsibilities: CDN, TLS termination, load balancing.
- API Gateway functions: authentication, rate limiting, quotas.
- Edge optimizations: caching, request shaping, edge compute.

5. Backend Core Services (10 min)

- BFF layer: payload aggregation and protocol translation.
- Domain services: encapsulated business logic and state.
- Service mesh for inter-service security, retries, and tracing.
- Communication patterns: synchronous RPC vs asynchronous events.

6. Data Storage & Retrieval (5 min)

- Core store types: SQL, key-value, document, search, time-series.
- CAP theorem and consistency vs availability trade-offs.

- Partitioning, replication, and backup strategies.

7. Scalability, Reliability & Performance (5 min)

- Vertical vs horizontal scaling approaches.
- Reliability techniques: health checks, auto-scaling, retries, circuit breakers.
- Performance levers: caching, backpressure mechanisms.
- Define and monitor SLIs/SLOs.

8. Deployment, Observability, Security & Ownership (as time allows)

- CI/CD pipelines and infrastructure-as-code.
- Deployment strategies: blue/green and canary releases.
- Observability foundations: metrics, logs, tracing.
- Security essentials: authentication/authorization and encryption.

9. Extensions, Trade-Offs, Bottlenecks & Evolution (as time allows)

- Event-driven vs request-driven design rationale.
- Multi-tenancy and tenant isolation highlights.
- Platformization and developer experience pointers.
- Testing & pre-production environment best practices.