educative

Solve Any System Design Interview Question

The 8-part **RESHADED** method:

1. Requirements
2. Estimation
3. Storage schema (optional)
4. High-level design
5. APIs
6. Detailed design
7. Evaluation
8. Distinctive component/feature

Step 1: Requirements
Gather functional & non-functional requirements

Consider:
- *System goals*
- *Key features*
- *System constraints*
- *User expectations*

Step 2: Estimation
Estimate hardware & infrastructure needed to implement at scale

Consider requirements for:
- *Number of servers*
- *Daily storage*
- *Network*

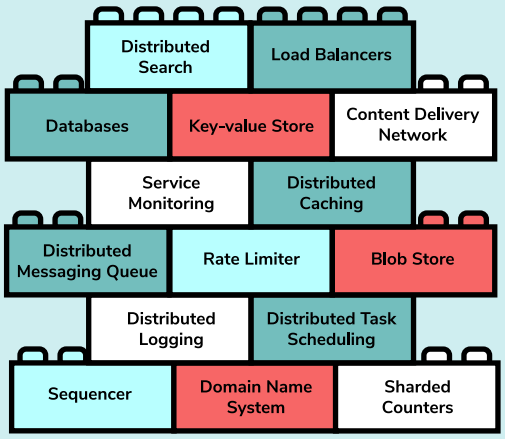
Step 3: Storage schema (optional)*
Articulate data model

Define:
- *Structure of data*
- *Tables to use*
- *Type of fields in tables*
- *Relationship between tables (optional)*

***Relevant when you:**
- *Expect highly normalized data*
- *Will store different parts of data in various formats*
- *Face performance & efficiency concerns around storage*

Step 4: High-level design
- Build high-level design
- Choose building blocks to meet functional requirements

For each, identify:
- ***How** they work*
- ***Why** they're needed*
- ***How** they integrate*



*This layered visual shows dependencies between building blocks. **Blocks in lower layers support those above.***

Step 5: APIs
Translate functional requirements into API calls

E.g.:
- ***Requirement:** Users should be able to access all items*
- ***API call:** GET / items*

Step 6: Detailed design
- *Improve high-level design*
- *Consider all non-functional requirements & complete design*

Step 7: Evaluation
- *Evaluate design against requirements*
- *Explain trade offs & pros/cons of different solutions*
- *Address overlooked design problems*

(8*) Distinctive component/feature
Discuss a distinctive feature that meets requirements
- *E.g. Concurrency control in high-traffic apps*

** Timing varies. Best done after completing design (E.g. Step 6 & 7)*

Building Blocks Glossary:

Domain Name System: *Maps domain names to IP addresses.*

Load Balancers: *Distributes client requests among servers.*

Databases: *Stores, retrieves, modifies, & deletes data.*

Key-Value Store: *Stores data as key-value pairs.*

Content Delivery Network: *Distributes in-demand content to end users.*

Sequencer: *Generates unique IDs for events & database entries.*

Service Monitoring: *Analyzes system for failures & sends alerts.*

Distributed Caching: *Stores frequently accessed data.*

Distributed Messaging Queue: *Decouples messaging producers from consumers.*

Publish-Subscribe System: *Supports asynchronous service-to-service communication.*

Rate Limiter: *Throttles incoming requests for services.*

Blob Store: *Stores unstructured data.*

Distributed Search: *Returns relevant content for user queries.*

Distributed Logging: *Enables services to log events.*

Distributed Task Scheduling: *Allocates resources to tasks.*

Sharded Counters: *Counts concurrent read/write requests.*