Solve Any System Design Interview Question

educative

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 and non-functional requirements

Consider:

- System goals
- Key features
- System constraints
- User expectations

STEP 2: Estimation

Estimates 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
- Types of fields in tables
- Relationships between tables (optional)

***Relevant when you:**

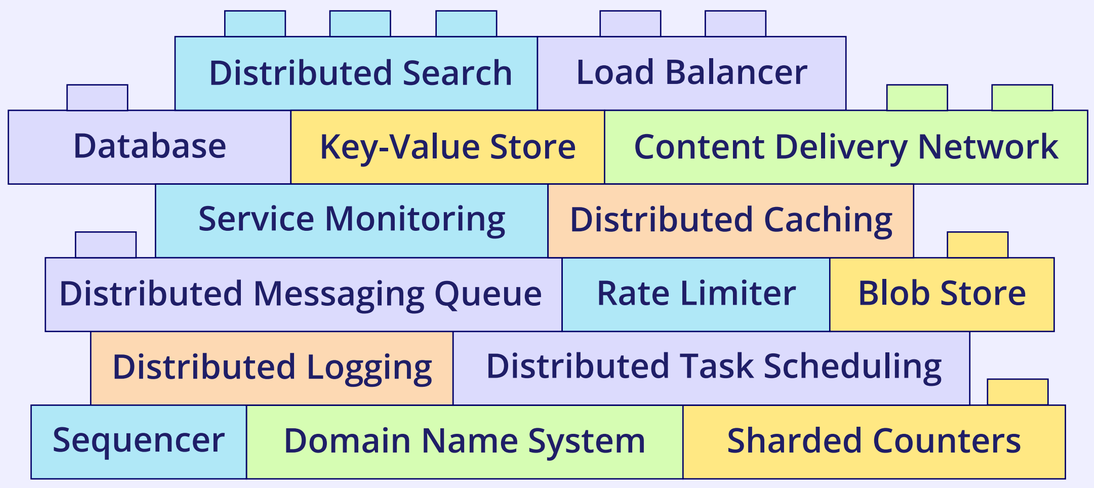
- Expect highly normalised data
- Will store different parts of data in various formats
- Face performance & efficiency concers around storage

STEP 4: High-level dsign

- 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
- Explains trade offs & pros and cons of different solutions
- Address overlooked design problems

STEP 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, and 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 and database entries

Servicing Monitoring
Analyzes system for failures and sends alerts

Distributed Caching
Decouples messaging producers from consumers

Distributed Messaging Queue
Decouples messaging producers from consumers

Publish-Scribe 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 readwrite requests