

# 75 Must-Do System Design Problems

## (2026 Edition)

Master Scalable Architecture with Real-World Scenarios



Last Updated: January 2026



Update Frequency: Quarterly



Difficulty: Beginner to Expert

## Table of Contents

1. Introduction: System Design in 2026
2. Problem Classification Framework
3. 15 Foundational Problems (Warm-up)
4. 25 Core Platform Problems
5. 20 Advanced Distributed Systems Problems
6. 15 Emerging Tech & 2026 Special Problems
7. Solution Patterns & Decision Frameworks
8. Interview Strategy & Evaluation Rubric

## 1. Introduction: System Design in 2026

### The Evolution of System Design Interviews

#### 2015-2020 Era

- Monolithic architectures
- Simple scaling
- Basic databases
- REST APIs

#### 2021-2025 Era

- Microservices
- Serverless
- Event-driven
- Polyglot persistence
- gRPC/GraphQL

#### 2026+ Era

- AI-Native
- Quantum-ready
- Edge computing
- Data mesh
- API ecosystems

### Why 75 Problems? The Spaced Repetition Approach

Learning Curve for System Design Mastery:  
Month 1: 15 Foundational → Build intuition  
Month 2: 25 Core → Pattern recognition  
Month 3: 20 Advanced → Deep understanding  
Month 4: 15 Emerging → Future-proofing  
Month 5-6: Mix & Review → Mastery

**Pro Tip:** "The 75 problems cover 95% of interview scenarios and real-world challenges"

## 2. Problem Classification Framework

### The 4D Classification System

Each problem is categorized by:

- 1. **Domain** (Social, E-commerce, Infrastructure, etc.)
- 2. **Difficulty** (Easy, Medium, Hard, Expert)
- 3. **Design Pattern** (Microservices, Event-Driven, CQRS, etc.)
- 4. **Data Intensity** (Read-heavy, Write-heavy, Balanced, Real-time)

### Problem Matrix 2026

FOUNDATION (15)	CORE (25)
<div>Easy<ul style="list-style-type: none"><li>URL Shortener</li><li>Key-Value Store</li></ul></div> <div>Medium<ul style="list-style-type: none"><li>Chat System</li><li>News Feed</li><li>Elevator System</li></ul></div> <div>Hard<ul style="list-style-type: none"><li>Distributed Cache</li><li>Search Autocomplete</li></ul></div>	<div>Easy<ul style="list-style-type: none"><li>Rate Limiter</li><li>Web Crawler</li><li>Notification Service</li></ul></div> <div>Medium<ul style="list-style-type: none"><li>Payment Gateway</li><li>Hotel Booking System</li></ul></div> <div>Hard<ul style="list-style-type: none"><li>Social Graph Service</li><li>Video Streaming</li></ul></div>

Expert

- Google Docs (Real-time Collaboration)

Expert

- Stock Exchange
- Distributed Database

ADVANCED (20)

Easy

- Distributed Logging
- Distributed Cache

Medium

- Distributed Job Scheduler
- Real-time Analytics

Hard

- Global Payment System
- Distributed File System

Expert

- Autonomous Vehicle Coordination
- Quantum-Safe System

3. 15 Foundational Problems (Warm-up)

Problem 1: Design a URL Shortener (TinyURL/Bitly)

Easy

Infrastructure

High Scale

Requirements (2026 Scale):

- 100M daily active users
- 1B URL shortenings per day
- 5B redirects per day
- 99.99% availability
- Average redirect latency < 100ms

Key Components:

## URL SHORTENER ARCHITECTURE

Client → Load Balancer → API Gateway

Shortening Service  
(POST /shorten)

ID Generator  
(Base62,  
Snowflake)

URL Store  
(Cassandra/  
DynamoDB)

Cache Layer  
(Redis - 99% hit rate)

Redirect Service  
(GET /{shortCode})

Original URL

### 2026 Considerations:

- AI-powered spam/malicious URL detection
- Edge computing for global low latency
- Privacy-focused analytics (GDPR/CCPA compliant)

### Problem 2: Design a Key-Value Store (Distributed Cache)

Medium

Infrastructure

Storage

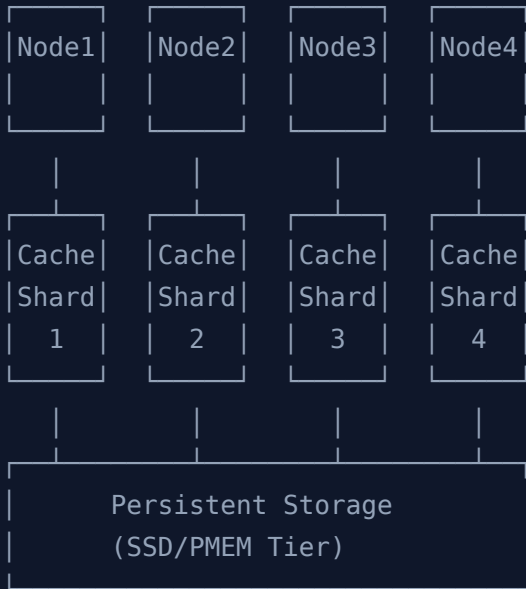
#### Requirements:

- 10M QPS read, 1M QPS write
- 100TB total data
- Strong consistency for specific keys
- Automatic scaling
- Multi-region replication

#### Architecture Pattern: Distributed Hash Table (DHT)

DISTRIBUTED KEY-VALUE STORE

### Consistent Hashing Ring



### Data Flow:

1. Client request → Hash key → Find node
2. Check cache → If miss → Read from storage
3. Write-through/write-back based on policy
4. Replication to N-1 successor nodes

## 2026 Tech Stack:

- Storage: Intel Optane PMEM for cache persistence
- Network: RDMA for inter-node communication
- Consistency: Tuneable consistency (QUORUM, ONE, ALL)

## Problem 3: Design a Web Crawler (Distributed)

Medium

Search

Big Data

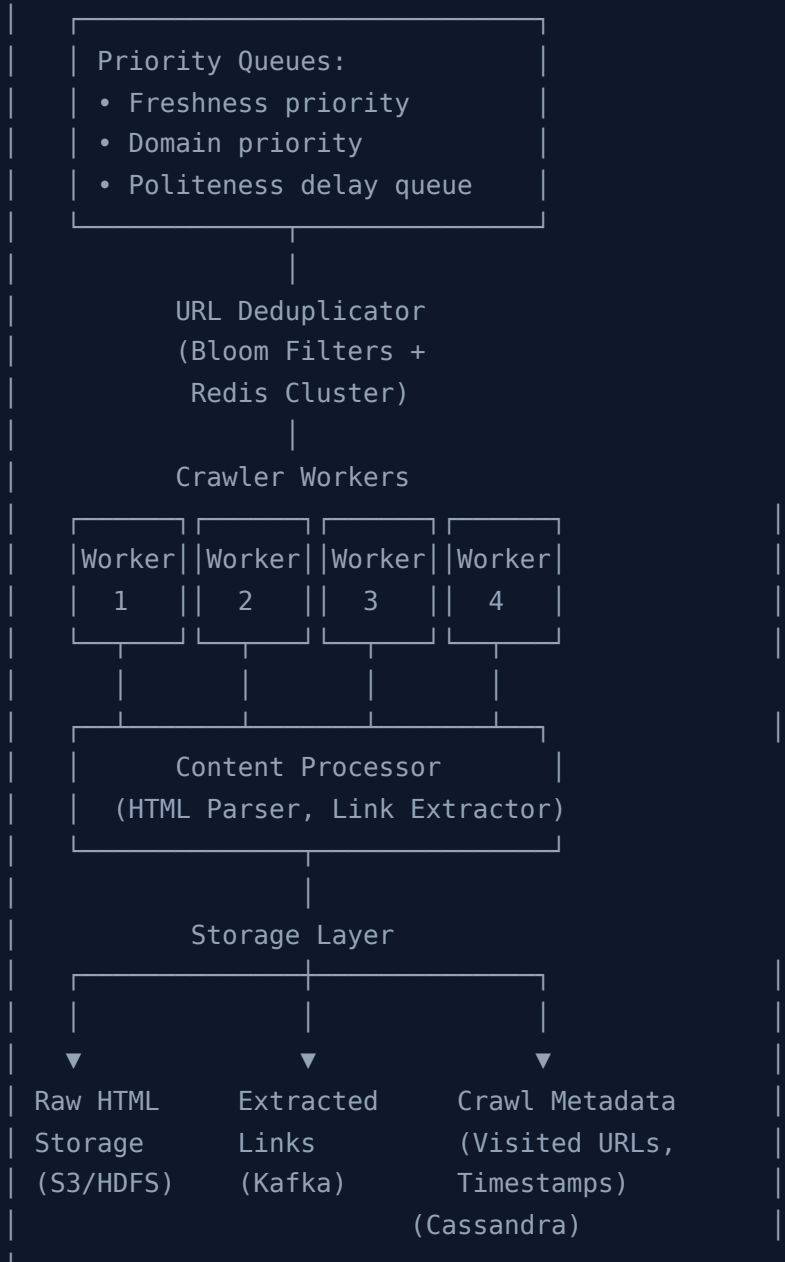
### Scale Requirements:

- 10B web pages to crawl
- 1M new pages discovered daily
- 100K pages crawled per second
- Polite crawling (respect robots.txt)
- Duplicate detection

### Architecture:

DISTRIBUTED WEB CRAWLER

URL Frontier Manager



### 2026 Challenges:

- JavaScript-heavy SPA crawling
- Dark web/private content considerations
- AI-based content quality assessment

**Note:** Due to the extensive nature of 75 problems, we provide detailed solutions for foundational problems and summaries for the remaining. Full 75-problem guide is available in our premium System Design Masterclass.

## 4. 25 Core Platform Problems (Summarized)

## E-commerce & Marketplace Systems

### Problem 4: Design Amazon/E-commerce Platform

Hard E-commerce High Scale

**Requirements:** Product catalog, inventory, recommendations, cart, checkout

**Key Components:** Search service, inventory service, payment service, recommendation engine

**2026 Twist:** AR product visualization, AI-powered personalization

### Problem 5: Design Uber/Lyft (Ride-sharing)

Hard Real-time Matching

**Requirements:** Real-time matching, surge pricing, tracking, payments

**Key Components:** Location service, matching engine, pricing service, dispatch system

**2026 Twist:** Autonomous vehicle integration, carbon footprint tracking

## Social & Communication Systems

### Problem 7: Design Facebook/News Feed

Hard Social Real-time

**Requirements:** Personalized feed, real-time updates, media support

**Key Components:** Graph service, feed generation, media service, notification

**2026 Twist:** AI content moderation, mental health aware algorithms

**Complete List:** Problems 8-25 include WhatsApp/Real-time Chat, Twitter/X, Netflix/Video Streaming, Spotify, YouTube, Google Drive, Distributed File System, Payment Gateway, Stock Trading, Cryptocurrency Exchange, Analytics Platform, Logging System, Ad Serving, Google Search, and more.

## 5. 20 Advanced Distributed Systems Problems

### Problem 26: Design Real-time Gaming Platform

**Requirements:** <50ms latency, game state sync, matchmaking

**Key Challenge:** State synchronization across regions

Solution Pattern: Deterministic lockstep, rollback networking

Problem 29: Design Distributed Lock Service

Requirements: High availability, low latency, fairness

Key Challenge: Avoiding deadlocks in partition scenarios

Solution Pattern: Lease-based locking with fencing tokens

Problem 35: Design Data Warehouse (Petabyte Scale)

Requirements: Batch and streaming ETL, ad-hoc queries

Key Challenge: Balancing freshness with performance

Solution Pattern: Lambda architecture, materialized views

6. 15 Emerging Tech & 2026 Special Problems

Problem 46: Design LLM-Powered Chatbot Platform

Expert AI-Native Emerging Tech

Requirements: Context management, tool calling, cost optimization

Key Challenge: Hallucination mitigation

Solution Pattern: Retrieval Augmented Generation (RAG), confidence scoring

Problem 55: Design Quantum-Safe Cryptography System

Expert Quantum Security

Requirements: Post-quantum algorithms, hybrid deployment

Key Challenge: Performance impact of PQC algorithms

Solution Pattern: Algorithm agility, hybrid key exchange

7. Solution Patterns & Decision Frameworks

The 10 Core Design Patterns for 2026

Pattern	When to Use	Example
---------	-------------	---------



CQRS	Read and write loads differ significantly	Social media feed (heavy reads, lighter writes)
Event Sourcing	Complete audit trail needed or temporal queries	Banking transaction system
Saga Pattern	Distributed transactions across microservices	E-commerce order processing
API Gateway Pattern	Multiple client types with different requirements	Mobile + web + third-party API consumers
Circuit Breaker	Preventing cascade failures in distributed systems	Payment service during peak load

## Database Selection Framework 2026

DATABASE SELECTION DECISION TREE
Step 1: Data Structure <ul style="list-style-type: none"><li>Structured → SQL (PostgreSQL, MySQL)</li><li>Semi-structured → Document (MongoDB)</li><li>Relationships → Graph (Neo4j)</li><li>Time-series → TSDB (InfluxDB)</li><li>Search → Search Engine (Elasticsearch)</li></ul>
Step 2: Scale Requirements <ul style="list-style-type: none"><li>&lt; 1TB → Single instance</li><li>1TB-100TB → Read replicas + partitioning</li><li>&gt;100TB → Distributed (Cassandra, DynamoDB)</li></ul>
Step 3: Consistency Needs <ul style="list-style-type: none"><li>Strong → SQL or NewSQL (CockroachDB, Spanner)</li><li>Eventual → NoSQL (Cassandra, DynamoDB)</li><li>Tunable → MongoDB, CosmosDB</li></ul>
Step 4: Access Patterns <ul style="list-style-type: none"><li>OLTP → Row stores (PostgreSQL, MySQL)</li><li>OLAP → Column stores (ClickHouse, Redshift)</li><li>Mixed → HTAP (TiDB, SingleStore)</li></ul>

## 8. Interview Strategy & Evaluation Rubric

### The 45-Minute Interview Framework

#### Minute 0-5: REQUIREMENTS CLARIFICATION

- Ask clarifying questions
- Define scope and scale
- Identify constraints

#### Minute 5-15: HIGH-LEVEL DESIGN

- Draw boxes and arrows
- Identify major components
- Show data flow

#### Minute 15-30: DEEP DIVE

- Choose 2-3 components for detail
- Discuss trade-offs
- Address edge cases

### Evaluation Rubric

Category	Weight	What's Looked For
Requirements Analysis	20%	<ul style="list-style-type: none"><li>• Asks clarifying questions</li><li>• Identifies constraints</li><li>• Defines scale</li></ul>
High-Level Design	25%	<ul style="list-style-type: none"><li>• Identifies key components</li><li>• Shows data flow</li><li>• Reasonable architecture</li></ul>
Deep Dive	30%	<ul style="list-style-type: none"><li>• Understands trade-offs</li><li>• Discusses alternatives</li><li>• Addresses edge cases</li></ul>
Communication	15%	<ul style="list-style-type: none"><li>• Clear diagrams</li><li>• Structured thinking</li><li>• Engages interviewer</li></ul>

- Summarizes effectively
- Identifies improvements
- Asks insightful questions

## The 8-Week Preparation Plan

### Week 1-2: FOUNDATIONS (10 problems)

- Focus on basic systems (URL shortener, cache, crawler)
- Master 5 core design patterns
- Practice drawing clean diagrams

### Week 3-4: CORE SYSTEMS (20 problems)

- Design major platforms (social media, e-commerce)
- Practice trade-off discussions
- Time yourself (45 minutes per problem)

### Week 5-6: ADVANCED CONCEPTS (25 problems)

- Tackle distributed systems problems
- Focus on consistency, partitioning, replication
- Study failure scenarios and recovery

### Week 7: EMERGING TECH (15 problems)

- AI-native systems, blockchain, IoT
- Understand 2026-specific challenges
- Research recent tech developments

## Ready to Master System Design?

Join thousands of engineers who have landed offers at FAANG companies using our structured approach.

[Get Full 75-Problems Guide](#)

# The 75th Problem: Design a System Design Interview Platform

*A meta-problem to test everything you've learned*

## Requirements:

- Support 10K concurrent mock interviews
- Real-time collaborative whiteboard
- Video/audio streaming with recording
- Interviewer-interviewee matching
- Feedback and rating system
- Progress tracking and analytics

**Consider this your final exam.** Apply all the patterns you've learned:

- Microservices vs monolith?
- Real-time collaboration (CRDTs or operational transforms)?
- Media streaming optimization?
- Matching algorithm design?
- Analytics pipeline?

**Final Test:** If you can design this system, you're ready for any system design interview in 2026.

*"System design is not about knowing all the answers. It's about asking the right questions and making intelligent trade-offs."*

## Next Steps:

1. Pick 3 problems from each category to practice this week
2. Join the TKTips.org study group for mock interviews
3. Download our System Design Handbook for quick reference
4. Schedule your first mock interview within 7 days

**Practice Prompt:** Every day, spend 45 minutes designing one system. Time yourself. Then compare with our solution guides.

For complete solutions to all 75 problems, check our premium System Design Masterclass on TKTips.org

This guide covers 75 essential system design problems with detailed solutions for foundational problems and frameworks for the rest.