System Design (HLD & LLD) Placement Guide

Author: Placements Simplified Editorial Team

Class Introduction: Architecting Distributed Systems

Welcome to the System Design Guide. When interviewing for mid-level software engineering roles or top-tier startup tracks, algorithms alone are not enough. You will be evaluated on your ability to design large-scale, fault-tolerant distributed systems. High-Level Design (HLD) covers system topology, scaling boundaries, database designs, caching strategies, and load balancing. Low-Level Design (LLD) evaluates code structure, SOLID principles, class boundaries, and reusable design patterns. This guide is structured into detailed, classroom-style lessons to prepare you for these interviews. Let's get started.

Lesson 1: Load Balancing

A load balancer is a component that distributes incoming user requests across multiple servers to ensure no single server gets overloaded. When traffic exceeds a server's capacity, performance drops or the system may fail, so load balancing helps maintain availability and efficiency. It acts as a traffic manager, improving throughput, reducing latency, and enabling scalability.

CORE BENEFITS:

• Distributes incoming requests across multiple servers to enable horizontal scaling and prevent overload.
• Improves system availability and performance by reducing response time and balancing traffic efficiently.

Real-World Example: When you open Netflix, millions of users send requests at the same time. A load balancer distributes these requests across multiple servers so that video streaming remains smooth without crashes.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?

A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 2: Caching & Invalidation

Caching stores frequently accessed data in faster memory (like RAM) for quick retrieval instead of repeatedly querying the database. It reduces database load and improves system performance. Commonly used data is kept readily available for faster access.

CORE BENEFITS:

• Reduces database load by minimizing repeated queries and lowering unnecessary data access operations.

• Improves response time and overall performance by fetching data from faster memory (RAM) and reducing network calls.

Real-World Example: Platforms like YouTube use caching and CDNs to store videos, images, and static content closer to users, ensuring faster loading and smooth playback.

Cache Invalidation & Consistency: Strategies like TTL (time-based expiry), write-through, or write-back ensure stale data is updated or removed. Systems must carefully balance freshness vs performance using eventual consistency, cache refresh policies, and controlled invalidation mechanisms.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?

A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 3: Proxies: Forward & Reverse

A proxy server acts as an intermediary between a client and a server, managing requests and responses to improve performance, security, and control. It can hide identities, optimize traffic flow, and add an extra layer between users and servers in a system.

CORE BENEFITS:
• Filters, logs, and transforms requests (e.g., headers, encryption, compression) while also optimizing and coordinating traffic.
• Enhances security by acting as a gatekeeper and hiding internal client/server details.

Forward Proxy vs Reverse Proxy: Forward Proxy acts on behalf of the client, handling requests and hiding the user's identity from the server (mainly used for privacy, access control, and filtering). Reverse Proxy acts on behalf of the server, managing incoming requests and distributing them across backend servers (helps improve performance, security, and scalability).

Real-World Example: When you access Netflix, a proxy layer may handle incoming requests, distribute traffic, and ensure smooth and secure streaming.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 4: CAP Theorem

CAP stands for Consistency, Availability, and Partition Tolerance. The theorem states that in a distributed system, you cannot fully achieve all three properties at the same time due to inherent trade-offs, so you must prioritize based on system requirements.

• **Consistency (C):** Means that any read request will return the most recent write. Data consistency is usually 'strong' for SQL databases and for NoSQL databases consistency may be anything from 'eventual' to 'strong'.
• **Availability (A):** Means that a non-responding node must respond in a reasonable amount of time. Most likely you will prefer a database with higher availability.
• **Partition Tolerance (P):** Means the system will continue to operate despite network or node failures.

Trade-offs: Since partition tolerance is essential, systems usually trade off between consistency (CP) and availability (AP). Availability is preferred for systems that must stay always online, while consistency is critical for financial or transactional systems. CA systems are only possible in single-node environments.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 5: Databases (SQL vs. NoSQL)

Databases store and manage application data. In system design interviews, you're often asked to design schemas, define primary keys, and choose the right storage type (SQL or NoSQL). A well-designed database ensures efficient data access and scalability.

• **Relational (SQL):** Structured tables with fixed schemas, primary/foreign keys, and ACID consistency constraints. Best for transaction-heavy financial services.
• **Non-Relational (NoSQL):** Flexible schemas (key-value, document, graph, column-family) that scale horizontally easily. Best for high-write social feeds.

Real-World Example: In Instagram, databases store user profiles, posts, comments, and relationships between users.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 6: Database Indexing

Indexing is a technique used to speed up data retrieval by creating a separate data structure that points to actual records. Instead of scanning entire tables, indexes allow faster lookups using efficient search methods.

CORE BENEFITS:

• Improves query performance by avoiding full table scans and using indexed access for faster data retrieval.

• Uses sorted data structures (like binary search) for quick searching but adds extra storage and slight overhead on write operations.

Example: In LinkedIn, indexing on user names or skills helps quickly search profiles from millions of records.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?

A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 7: Database Replication

Replication involves creating multiple copies of a database to improve availability and handle high traffic. If one database fails, others can continue serving requests, ensuring system reliability.

CORE BENEFITS:
• Enhances availability and fault tolerance while supporting read scaling through the use of replicas.
• Replication can be synchronous (strong consistency) or asynchronous (better performance depending on system needs).

Example: In Facebook, replication ensures users can still access data even if some database servers go down.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 8: Database Sharding

Sharding is the process of splitting a large database into smaller pieces (shards) distributed across multiple servers. Each shard handles a portion of the data, helping scale systems horizontally.

CORE BENEFITS:

• Distributes data across multiple servers to handle massive scale and reduce load on a single database.
• Requires careful shard key selection to ensure balanced distribution and efficient querying.

Example: In Twitter, user data is sharded across multiple servers so the platform can handle millions of active users efficiently.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?

A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 9: Database Partitioning

Partitioning is the process of dividing a large database into smaller parts within the same server to improve query performance and manageability. It helps in organizing data efficiently without distributing it across multiple machines.

CORE BENEFITS:

• Improves query performance by scanning smaller, more targeted data chunks instead of the entire dataset.
• Keeps data logically separated (e.g., by date or region) while operating within a single database or server.

Example: In Twitter, tweets may be partitioned by date so recent data can be accessed faster.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?

A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 10: Horizontal vs. Vertical Scaling

Scaling is the technique of adding capacity to systems to meet traffic spikes:
• Vertical Scaling: Upgrading memory (RAM) or processors (CPU) of a single server. It has hard hardware limits and creates a Single Point of Failure (SPOF).
• Horizontal Scaling: Adding more server machines. Traffic is distributed using Load Balancers. It is infinitely scaleable but adds network complexity and consistency issues.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 11: Content Delivery Networks (CDNs)

CDNs cache static assets (images, videos, JS files) on edge servers close to users, reducing request latency and server load.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 12: Consistent Hashing Systems

Consistent hashing maps request keys and servers to a circular ring, minimizing key relocations when server nodes are added or removed.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 13: Message Queues: Kafka vs. RabbitMQ

Message queues decouple service processing. Kafka (log-based, high throughput, persistent) vs. RabbitMQ (AMQP broker, complex routing).

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 14: API Gateways & Rate Limiting

The entry point routing traffic, handling JWT authentication, logging metrics, and implementing rate limiting (token bucket).

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 15: DNS: Domain Name System Routing

Translates domain names to IP addresses. Uses geographical routing, Anycast routing, and TTL caches to direct users to nearby servers.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 16: Distributed Lock Managers (Redlock)

Coordinates resource access across instances using Redis (Redlock) or ZooKeeper, preventing concurrent write collisions.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 17: Distributed Consensus: Paxos and Raft

Consensus algorithms ensure multiple node replicas agree on system state. Raft uses leader election to simplify Paxos states.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 18: Heartbeats & Gossip Protocols

Heartbeats check server node liveness. Gossip protocol propagates node status updates across large clusters asynchronously.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 19: MapReduce & Batch Processing

Splits massive data processing tasks across server clusters using Map (filter/sort data) and Reduce (summarize outputs) stages.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 20: WebSockets vs. Long Polling

WebSockets create persistent, bi-directional TCP connections for real-time applications (chat, notifications) instead of polling APIs.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 21: Vector Databases: Pinecone & Milvus

Specialized databases for AI. Store and query vector embeddings using high-dimensional nearest-neighbor searches.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?

A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 22: SOLID: Single Responsibility Principle

A class should have one, and only one, reason to change. Each class should focus on a single core responsibility.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?

A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 23: SOLID: Open-Closed Principle

Software entities should be open for extension but closed for modification. Extend class functionality via polymorphism and interfaces.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 24: SOLID: Liskov Substitution Principle

Subclasses must be substitutable for their parent classes without breaking application correctness.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 25: SOLID: Interface Segregation Principle

Clients should not be forced to implement interface methods they do not use. Prefer small, focused interfaces.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?

A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 26: SOLID: Dependency Inversion Principle

Depend on abstractions (interfaces) rather than concrete implementations (classes). Enable dependency injection.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 27: Design Patterns: Singleton Pattern

Restricts a class to a single global instance. Used for Database connection pools and application configurations.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 28: Design Patterns: Factory Method Pattern

Provides an interface for creating objects in a superclass, but lets subclasses alter the type of created objects.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 29: Design Patterns: Abstract Factory Pattern

Creates families of related objects without specifying their concrete classes (e.g., UI component factory).

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 30: Design Patterns: Builder Pattern

Separates the construction of a complex object from its representation, allowing step-by-step assembly.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 31: Design Patterns: Prototype Pattern

Creates new objects by copying an existing instance (cloning), avoiding expensive database instantiation runs.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 32: Design Patterns: Adapter Pattern

Allows incompatible interfaces to work together by wrapping a class in an adapter interface wrapper.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 33: Design Patterns: Decorator Pattern

Dynamically adds behavior and responsibilities to an object without subclassing (e.g., input stream wrappers).

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 34: Design Patterns: Facade Pattern

Provides a simplified interface to a complex subsystem of classes, reducing system coupling.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 35: Design Patterns: Proxy Pattern

Provides a surrogate or placeholder object to control access to the original object (e.g. lazy loading, caching).

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 36: Design Patterns: Observer Pattern

Establishes a one-to-many relationship so when one object changes state, all its observers are notified automatically.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 37: Design Patterns: Strategy Pattern

Defines a family of algorithms, encapsulates each one, and makes them interchangeable at runtime (e.g., payment options).

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 38: Design Patterns: Template Method Pattern

Defines the skeleton of an algorithm in a method, deferring some steps to subclasses without modifying overall structure.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?

A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 39: Design Patterns: Command Pattern

Encapsulates a request as an object, letting you parameterize clients with queues, logs, and undoable operations.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?

A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 40: Design Patterns: State Pattern

Allows an object to alter its behavior when its internal state changes (e.g., order processing flow state).

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 41: Case Study: Design a URL Shortener (TinyURL)

Requires designing hash generation algorithms (MD5, Base62), database design for user mappings, and caching redirections.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 42: Case Study: Design a Rate Limiter

Requires designing system filters to block spam traffic using Redis, Token Bucket, or Sliding Window algorithms.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 43: Case Study: Design a Web Crawler

Requires designing high-scale HTML download workers, URL deduplication filters (Bloom filters), and queue systems.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 44: Case Study: Design Twitter Feed Delivery

Requires trading off Push (fan-out on write for active users) vs. Pull (fan-out on read for celebrity users) caching models.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 45: Case Study: Design YouTube/Netflix Streaming

Requires designing video transcoders, chunking strategies (DASH/HLS).

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.

Lesson 46: Case Study: Design Uber/Ride Sharing

Requires designing geo-spatial indexes (Google S2, Uber H3 grids) to match riders with drivers in real-time.

■ High-Yield Interview Q&A:

Q: How do you discuss this concept in a system design interview?
A: In a system design interview, do not just define the term. Walk the interviewer through the trade-offs. For instance, explain when you would choose consistency over availability under network partitions, or why you would trade database writes for faster reads using caches. Frame it based on the constraints of the system you are building.

■ Classroom Note & Analogy:

Think of this architecture like organizing a large-scale event. You need managers, coordinators, and communication buffers to prevent bottleneck stampedes at the gates. Clean structures ensure a smooth flow.