

# CheatSheet: Feature Design For Job Interview

## INTERVIEW

- PDF Link: [cheatsheet-featuredesign-A4.pdf](#), Category: interview
- Blog URL: <https://cheatsheet.dennyzhang.com/cheatsheet-featuredesign-A4>
- Related posts: CheatSheet: Leetcode For Code Interview, CheatSheet: System Design For Job Interview, #denny-cheatsheets

File me Issues or star this repo.

## 1.1 Top 20 Design Problems For Technical Modules

| Num | Name   | Summary          |
|-----|--|------------------|
| 1   | Design a distributed counter                               | link, link       |
| 2   | Delayed task scheduling                                    | link             |
| 3   | Design a thread-safe Hashmap                               | link, link       |
| 4   | Design a distributed Hashmap                               |                  |
| 5   | Design a distributed UUID generator                        |                  |
| 6   | Design An API Rate Limiter                                 | link, link, link |
| 7   | Design a distributed transaction                           |                  |
| 8   | Top URL hits   |                  |
| 9   | Unique url hits  |                  |
| 10  | Design a distributed transactions                          |                  |
| 11  | Design a load balancer                                     |                  |
| 12  | Design a client-server API to build a rich document editor |                  |
| 13  | Design online/offline status system                        |                  |
| 14  | Design a circuit breaker                                   |                  |
| 15  | Design a secrets management system                         |                  |
| 16  | Design data sync for a distributed system                  |                  |
| 17  | Design: A Parking Lot Service                              | link             |
| 18  | Design: A URL Redirecting Feature                          |                  |
| 19  | Design a service auto-discovery feature                    |                  |
| 20  | Design A big file transfer feature                         |                  |
| 21  | TODO   |                  |

## 1.2 Top 30 Concepts For Feature/System Design

| Num | Name                                    | Summary   |
|-----|---|---|
| 1   | Caching                                 | Stores data so that future requests of data retrieval can be faster |
| 2   | Message Queue                           | Provides an asynchronous communications protocol,                   |
| 3   | Data Partition & Sharding               | Break up a big data volume into many smaller parts                  |
| 4   | DB Indexing                             | Create indexes on multiple columns to speed up table look up        |
| 5   | DB replication                          | Duplicate data to increase service availability                     |
| 6   | CAP: Consistency/Availability/Partition | A distributed database system can only have 2 of the 3              |
| 7   | DB: SQL & NoSQL                         | Relational databases and non-relational databases                   |
| 8   | Concurrency & Communication             |   |
| 9   | Networking: HTTP                        |   |
| 10  | Pull vs Push model                      |   |
| 11  | Consistency Module                      | weak consistency, eventual consistency, strong consistency          |
| 12  | Conflict resolution                     | Quorum, vector lock, reconcile on read/write                        |
| 13  | Garbage Collection                      |   |
| 14  | Memory Management                       |   |
| 15  | Heartbeats                              |   |
| 16  | Self Protection                         | API Rate limit, Circuit breaker, bulkhead, throttling               |
| 17  | Filesystem                              |   |
| 18  | API: gRPC vs REST                       |   |
| 19  | Load balancer                           |   |
| 20  | Scale up vs Scale out                   | Vertical scaling and Horizontal scaling                             |
| 21  | API Design                              |   |
| 22  | Pessimistic And Optimistic Locking      |   |
| 23  | Session management                      |   |
| 24  | Networking: TCP vs UDP                  |   |
| 25  | Consistency patterns                    | Weak consistency, Eventual consistency, Strong consistency          |
| 26  | Availability patterns                   | Fail-over vs Replication  |
| 27  | CDN - Content Delivery Network          | Edge caching  |
| 28  | Monitoring                              |   |
| 29  | Security                                |   |
| 30  | Networking: DNS                         |   |

## 1.3 Advanced Data Structure & Algorithms

| Num | Name                                     | Summary                  |
|-----|--|--------------------------|
| 1   | Consistent Hash                          |                          |
| 2   | Delayed queue                            | Run scheduled tasks      |
| 3   | Two-phase commit/Three-phase commit      |                          |
| 4   | SSTable (Sorted Strings Table)           |                          |
| 5   | LSM (Log Structured Merge Trees)         |                          |
| 6   | B+ Tree                                  |                          |
| 7   | Gossip                                   | Propagate cluster status |
| 8   | CRDTs (Consistent Replicated Data Types) |                          |
| 9   | Vector Clocks/Version Vectors            |                          |
| 10  | Paxos and raft protocol                  |                          |
| 11  | Merkle Tree                              |                          |
| 12  | Bloom filter                             |                          |

<https://raw.githubusercontent.com/dennyzhang/cheatsheet.dennyzhang.com/master/cheatsheet-featuredesign-A4/dynamo-summary.png>

## 1.4 Explain workflow: What happens when XXX?

| Num | Name                                  | Summary |
|-----|---------------------------------------|---------|
| 1   | When happens when I search in google? |         |
| 2   | How loadbalancer works                |         |
| 3   | Explain three phase commit model      |         |
| 4   | Explain HTTP return code              |         |
| 5   | Explain Mysql DB replication model    |         |
| 6   | Explain gossip protocol               |         |
| 7   | Explain CAP                           |         |
| 8   | Explain Hadoop file system            |         |

## 1.5 Explain tools: how XXX supports XXX?

| Num | Name                                  | Summary |
|-----|---------------------------------------|---------|
| 1   | How JDK implement hashmap?            |         |
| 2   | Explain java garbage collection model |         |
| 3   | Explain raft/etcd                     |         |
| 4   | How OS supports XXX?                  |         |

## 1.6 Cloud Design Principles

| Num | Name                       | Summary                 |
|-----|----------------------------|-------------------------|
| 1   | Fail fast                  |                         |
| 2   | Design for failure         |                         |
| 3   | Immutable infrastructure   |                         |
| 4   | Cats vs Cattle             | Avoid snowflake servers |
| 5   | Auto healing               |                         |
| 6   | Async programming          |                         |
| 7   | GitOps operational model   |                         |
| 8   | Event-Driven Architectures |                         |

## 1.7 Cloud Design Patterns

| Num | Name                        | Summary  |
|-----|-----------------------------|--|
| 1   | Ambassador pattern          | Create helper service to send network requests, besides the main service |
| 2   | Cache-Aside pattern         | Load data on demand into a cache from a data store                       |
| 3   | Circuit Breaker pattern     | If a request takes too many resource, abort it                           |
| 4   | Bulkhead pattern            | Isolate elements into pools, so that one fire won't burn all             |
| 5   | Gateway Aggregation pattern | Aggregate multiple individual requests into a single request             |
| 6   | Priority Queue pattern      | Support different SLAs for different individual clients                  |
| 7   | Strangler pattern           | Incrementally migrate a legacy system piece by piece                     |

## 1.8 Misc

| Num | Name  | Summary                           |
|-----|---|-----------------------------------|
| 1   | How to store 2TB data into 3 disks of 1TB. And be tolerant for one disk failure | A, B, C. And C = A XOR B          |
| 2   | Find out the difference between two files. Majority of these two are the same   | #lcs - Longest Common Subsequence |
| 3   | How to support feature of "diff 1.txt 2. txt"                                   |                                   |
| 4   | Avoid double payment in a distributed payment system                            | link                              |

## 1.9 Top 20 Object-Oriented Design Problems

| Num | Problem                         | Category/Tag           | Example   |
|-----|---------------------------------|------------------------|---|
| 1   | Cache                           | #linkedlist, #oodesign | Leetcode: LRU Cache, Leetcode: LFU Cache, Leetcode: All O(1) Data Structure |
| 2   | Throttling                      | #linkedlist, #oodesign | Leetcode: Design Hit Counter, Leetcode: Logger Rate Limiter                 |
| 3   | Iterator                        | #oodesign              | Leetcode: Binary Search Tree Iterator, Leetcode: Design Circular Iterator   |
| 4   | Design Log Storage System       | #oodesign              | Leetcode: Design Log Storage System   |
| 5   | Linked List with random access  | #oodesign              | Leetcode: Design Linked List  |
| 6   | Max Stack                       | #stack, #oodesign      | Leetcode: Max Stack   |
| 7   | Design HashMap                  | #oodesign              | Leetcode: Design HashMap  |
| 8   | Circular Queue                  | #oodesign              | Leetcode: Design Circular Queue, Leetcode: Design Circular Deque            |
| 9   | Trie tree                       | #oodesign              | Leetcode: Implement Trie (Prefix Tree)                                      |
| 10  | Get Median                      | #oodesign              | Leetcode: Find Median from Data Stream                                      |
| 11  | Range Sum Query                 | #oodesign              | Leetcode: Range Sum Query - Immutable, Leetcode: Range Sum Query - Mutable  |
| 12  | Design File System              | #oodesign              | Leetcode: Design File System  |
| 13  | Insert Delete GetRandom O(1)    | #oodesign, #random     | Leetcode: Insert Delete GetRandom O(1)                                      |
| 14  | Insert Delete GetRandom O(1) II | #oodesign, #random     | Leetcode: Insert Delete GetRandom O(1) - Duplicates allowed                 |

## 1.10 More Resources

License: Code is licensed under MIT License.

<https://github.com/donnemartin/system-design-primer>

<https://github.com/checkcheckzz/system-design-interview>

<https://github.com/binhnguyennus/awesome-scalability>

<https://docs.microsoft.com/en-us/azure/architecture/patterns/>