

# SYSTEM DESIGN PLAYLIST

## HIGH LEVEL DESIGN-

FUNCTIONAL REQUIREMENTS

NON FUNCTIONAL REQUIREMENTS

### Step1: Fundamentals

- Serverless vs Serverful
- Horizontal vs Vertical Scaling
- What are threads?
- What are pages?
- How does the internet work?

### Step2: Databases

- SQL vs NoSQL Dbs
- In-memory DBs
- Data Replication & Migration
- Data Partitioning
- Sharding

### Step3: Consistency vs Availability

- Data Consistency & its levels
- Isolation & its levels
- CAP Theorem

### Step4: Cache

- What is Cache? (Redis, Memcached)
- Write Policies: write back, through & around
- Replacement Policies: LFU, LRU, Segmented LRU etc.
- Content Delivery Networks (CDNs)

### Step5 : Networking

- TCP vs UDP
- What is http (1/2/3) & https
- Web sockets
- WebRTC & video streaming

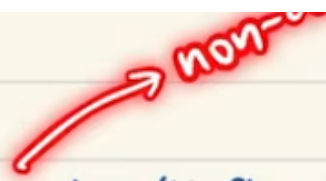
### Step6 : Load Balancers

- Load Balancing Algorithms (Stateless & Stateful)
- Consistent Hashing
- Proxy & Reverse Proxy
- Rate Limiting



### Step7 : Message Queues

- Asynchronous Processing (Kafka, RabbitMQ)
- Publisher-Subscriber Model



### Step8 : Monoliths vs Microservices

- Why Microservices?
- Concept of 'Single Point of Failure'
- Avoiding Cascading Failures
- Containerization (Docker)
- Migrating to Microservices

### Step9 : Monitoring & Logging

- Logging events & monitoring metrics
- Anomaly Detection

### Step10 : Security

- Tokens for auth
- SSO & OAuth
- Access Control Lists & Rule Engines
- Encryption

## Step 11 : System Design Tradeoffs

- Push vs Pull architecture
- Consistency vs Availability
- SQL vs NoSQL Dbs
- Memory vs Latency
- Throughput vs Latency
- Accuracy vs Latency

## Step 12 : practice, practice, practice

1. Youtube
2. Twitter
3. Whatsapp
4. Uber
5. Amazon
6. Dropbox/Google Drive
7. Netflix
8. Instagram
9. Zoom
10. Booking.com/ Airbnb

LOW LEVEL DESIGN-

### Step1: Object Oriented Programming

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism
- SOLID Principles

### Step2: Design Patterns

- Creational (singleton, factory etc.)
- Structural (Proxy, Bridge etc.)
- Behavioral (Strategy, Command, Observer etc.)

### Step3: Concurrency & Thread safety

- Thread safe injection
- Locking mechanisms
- producer-consumer
- race conditions & synchronization

### Step4: UML Diagrams

## Step5: APIs

- API design
- req/res object modeling
- versioning & extensibility
- Clean Code Principles: DRY, SRP etc.
- Avoiding God classes

## Step6: Common LLD Problems

1. Design a Tic-tac-toe or chess game
2. Design a Splitwise App
3. Design a Parking lot
4. Design an Elevator System with multiple lifts
5. Design a Notification System
6. Design a Food delivery app
7. Design a Movie ticket booking system
8. Design a URL shortener
9. Design a Logging framework
10. Design a Rate Limiter