System Design – 100+ Lab Exercises (Basic, Intermediate, Advanced)

□ Basic Level (30+ Exercises)

Objective: Understand foundational system design principles, components, and communication models.

□ Core Concepts

- Define **monolithic vs microservices** architecture with examples.
- Design a simple **URL shortener**.
- Model a **basic CRUD API** with REST.
- Explain **client-server architecture**.
- Simulate **stateless vs stateful services**.
- Explore **Synchronous vs Asynchronous communication**.

□ System Components

- Draw and label components of a **web application stack** (frontend, backend, DB).
- Diagram a **three-tier architecture** (presentation, logic, storage).
- Design a **simple authentication system** with sessions.
- Identify bottlenecks in a naive **single-server** system.

□ Networking & Protocols

- Design a TCP-based **chat application** prototype.
- Simulate **load balancing** with round-robin algorithm.
- Compare **HTTP vs WebSocket** for live updates.
- Build a simple **RESTful API** with JSON payloads.

□ Storage & Database Basics

- Choose between **SQL vs NoSQL** for a blog system.
- Model ER diagram for a **task management system**.
- Implement **database sharding simulation**.
- Design a **read replica architecture**.

□ Intermediate Level (40+ Exercises)

Objective: Architect scalable, resilient systems with caching, messaging, and horizontal scaling.

□ Caching & CDN

- Integrate **Redis** for caching API responses.
- Simulate a **cache eviction policy** (LRU, LFU).
- Design a **CDN** for static content delivery.
- Discuss **write-through vs write-back caching**.

□ Messaging & Queues

- Build a **job queue** using RabbitMQ/Kafka.
- Simulate **asynchronous order processing**.
- Design **rate limiting** using token bucket algorithm.
- Implement a **notification service** with pub-sub model.

□ Scalability & Load

- Design a **horizontal scaling strategy** for a file upload service.
- Simulate **auto-scaling with load spikes**.
- Build a **health check system** for microservices.
- Implement a **circuit breaker pattern**.

□ Security & Access

- Design an **OAuth 2.0 flow** for third-party login.
- Simulate **JWT token verification**.
- Create **role-based access control (RBAC)** model.
- Secure an API using **API gateway and throttling**.

□ Data Consistency & Availability

- Design **eventual consistency** for a replicated DB.
- Implement a **distributed lock mechanism**.
- Use **quorum-based reads/writes**.
- Compare **CAP theorem trade-offs**.

□ Advanced Level (40+ Exercises)

Objective: Architect distributed, fault-tolerant systems for large-scale and real-time applications.

□ System Design Problems

- Design **YouTube-like video streaming service**.
- Design **WhatsApp messaging system**.
- Architect **Uber ride-matching system**.
- Design a **real-time multiplayer game backend**.
- Design **Instagram backend** with photo delivery and follower feed.
- Create **scalable search engine** architecture.

□ Distributed Systems

- Simulate **Leader election** in distributed nodes.
- Implement **consistent hashing** for partitioning.
- Build **event sourcing** pipeline.
- Explain **Paxos vs Raft** consensus protocols.

□ Data Engineering & Analytics

- Design a **real-time analytics system**.
- Build a **data lake and ETL pipeline**.
- Create a **log aggregation system** (e.g., ELK stack).
- Design a **dashboard for monitoring 10k+ services**.

□ High Availability

- Implement **failover mechanism**.
- Design **multi-region deployment** strategy.
- Discuss **RTO vs RPO** in disaster recovery.
- Build a **global load balancer architecture**.

□ Observability

- Implement **log tracing** across microservices.
- Design a **metrics monitoring stack** (Prometheus + Grafana).
- Build **alerting mechanisms** with SLO thresholds.
- Use **distributed tracing (Jaeger/OpenTelemetry)**.

□ Capstone Projects (Choose 2-3)

- **Design a Scalable Food Delivery Platform** (Swiggy/Zomato style)
- **Build a Scalable Video Conferencing System** (Zoom style)
- **Create a High-Frequency Trading Platform** (Stock ticker simulator)
- **Design an IoT Data Ingestion Platform** (MQTT + Cloud + Analytics)
- **Build a Scalable Microservices E-commerce System**

□ Tools & Technologies

- **Load Testing:** Apache JMeter, k6
- **Architecture Diagrams:** Draw.io, Lucidchart
- **Infrastructure as Code:** Terraform, AWS CDK
- **Containerization & Orchestration:** Docker, Kubernetes
- **CI/CD Tools:** Jenkins, GitHub Actions, ArgoCD