Complete System Design Curriculum: Basics to Advanced

Part 1: Foundation Concepts

Chapter 1: Introduction to System Design

- What is System Design and why it matters
- Difference between software design and system design
- System design interview expectations
- Real-world system design examples

Chapter 2: Computer Architecture Basics

- How computers work: CPU, memory, storage
- Understanding latency and throughput
- Network basics: bandwidth, protocols (TCP/IP, HTTP, WebSockets)
- Client-server architecture

Chapter 3: Scalability Fundamentals

- Vertical scaling vs horizontal scaling
- Understanding bottlenecks
- Performance metrics (latency, throughput, availability)
- Back-of-the-envelope calculations

Part 2: Core Building Blocks

Chapter 4: Load Balancing

- What is load balancing and why we need it
- Load balancing algorithms (Round Robin, Least Connections, IP Hash)
- Layer 4 vs Layer 7 load balancing
- Health checks and failure detection
- Popular load balancers: NGINX, HAProxy, AWS ELB

Chapter 5: Caching

- What is caching and when to use it
- Cache hit vs cache miss
- Caching strategies: Cache-aside, Write-through, Write-back, Refresh-ahead

- Cache eviction policies: LRU, LFU, FIFO
- Cache invalidation challenges
- Distributed caching: Redis, Memcached
- CDN (Content Delivery Networks)

Chapter 6: Database Fundamentals

- SQL vs NoSQL databases
- ACID properties
- Database indexes and how they work
- Normalization vs denormalization
- Connection pooling

Chapter 7: Database Scaling

- Read replicas and replication lag
- Master-slave vs master-master replication
- Database sharding (horizontal partitioning)
- Sharding strategies: Hash-based, Range-based, Directory-based
- Consistent hashing
- Database partitioning techniques

Chapter 8: Storage Systems

- Block storage vs Object storage vs File storage
- Distributed file systems (HDFS, GFS)
- Blob storage (Amazon S3, Azure Blob)
- Database types: Relational, Document, Key-Value, Graph, Wide-Column

Part 3: Communication & Messaging

Chapter 9: APIs and Communication Protocols

- REST API design principles
- GraphQL basics
- gRPC and Protocol Buffers
- WebSockets for real-time communication
- API versioning strategies

- Rate limiting and throttling

Chapter 10: Message Queues

- Synchronous vs asynchronous communication
- Message queue fundamentals
- Queue vs Topic (Pub/Sub)
- Popular message queues: RabbitMQ, Apache Kafka, Amazon SQS
- Message ordering and delivery guarantees
- Dead letter queues

Chapter 11: Event-Driven Architecture

- Event sourcing
- CQRS (Command Query Responsibility Segregation)
- Event streaming platforms
- Real-time data processing

Part 4: System Reliability

Chapter 12: High Availability

- Understanding the nine nines (99.9%, 99.99%, etc.)
- Single points of failure
- Redundancy and replication
- Failover mechanisms
- Active-active vs active-passive setups

Chapter 13: Fault Tolerance and Resilience

- Graceful degradation
- Circuit breaker pattern
- Retry logic and exponential backoff
- Bulkhead pattern
- Chaos engineering principles

Chapter 14: Monitoring and Observability

- Logging best practices
- Metrics collection and monitoring

- Distributed tracing
- Alerting strategies
- Popular tools: Prometheus, Grafana, ELK Stack, Datadog

Part 5: Advanced Concepts

Chapter 15: Microservices Architecture

- Monolith vs Microservices
- Service decomposition strategies
- Inter-service communication
- API Gateway pattern
- Service mesh (Istio, Linkerd)
- Microservices challenges and trade-offs

Chapter 16: Distributed Systems Theory

- CAP theorem (Consistency, Availability, Partition tolerance)
- BASE vs ACID
- Eventual consistency
- Strong consistency vs eventual consistency trade-offs
- Distributed transactions and Two-Phase Commit
- Consensus algorithms: Paxos and Raft

Chapter 17: Data Processing at Scale

- Batch processing vs Stream processing
- MapReduce paradigm
- Apache Hadoop ecosystem
- Apache Spark
- Lambda and Kappa architectures
- Real-time analytics

Chapter 18: Search Systems

- Full-text search fundamentals
- Inverted indexes
- Ranking and relevance

- Elasticsearch and Apache Solr
- Search optimization techniques

Chapter 19: Distributed Caching and Data Stores

- Distributed cache coherence
- Write-through and write-behind caching at scale
- Session management in distributed systems
- Data locality and hotspot management

Chapter 20: Security in System Design

- Authentication vs Authorization
- OAuth 2.0 and JWT
- SSL/TLS and encryption
- DDoS protection
- API security best practices
- Rate limiting and API keys
- Security at different layers

Part 6: Specialized Topics

Chapter 21: Real-Time Systems

- Designing chat applications
- Live streaming architecture
- Real-time notifications
- WebRTC for video/audio
- Presence systems

Chapter 22: Content Delivery

- CDN architecture deep dive
- Video streaming systems
- Image optimization and serving
- Progressive loading

Chapter 23: Location-Based Services

- Geospatial indexes

- Proximity searches
- Geohashing and QuadTrees
- Map services design

Chapter 24: Recommendation Systems

- Collaborative filtering
- Content-based filtering
- Hybrid approaches
- Real-time vs batch recommendations

Chapter 25: Rate Limiting and Traffic Management

- Token bucket algorithm
- Leaky bucket algorithm
- Fixed window vs Sliding window
- Distributed rate limiting

Part 7: Design Patterns and Best Practices

Chapter 26: Common Design Patterns

- Proxy pattern
- Adapter pattern
- Facade pattern
- Observer pattern
- Saga pattern for distributed transactions
- Strangler fig pattern for migrations

Chapter 27: Performance Optimization

- Database query optimization
- N+1 query problem
- Connection pooling
- Compression techniques
- Lazy loading vs eager loading

Chapter 28: Scalability Patterns

- Read-heavy vs write-heavy optimizations

- Hot partition handling
- Data denormalization strategies
- Materialized views

Part 8: Practical System Design

Chapter 29: Design Methodology

- Requirements gathering (functional and non-functional)
- Capacity estimation
- API design
- Database schema design
- High-level architecture
- Detailed component design
- Bottleneck identification

Chapter 30: Common System Design Examples

- URL Shortener (like bit.ly)
- Social Media Feed (like Twitter/Facebook)
- Photo/Video Sharing (like Instagram/YouTube)
- Messenger System (like WhatsApp)
- Ride-Sharing App (like Uber)
- E-commerce Platform (like Amazon)
- Payment System
- Search Engine (like Google)
- Web Crawler
- Collaborative Editing (like Google Docs)
- Notification System
- Rate Limiter
- Distributed Cache
- Ticketing System (like Ticketmaster)

Part 9: Modern Architecture

Chapter 31: Cloud-Native Architecture

- Containerization (Docker)
- Container orchestration (Kubernetes)
- Serverless computing
- Functions as a Service (FaaS)
- Cloud design patterns

Chapter 32: Data Pipeline Architecture

- ETL vs ELT
- Data lakes vs Data warehouses
- Stream processing pipelines
- Data quality and validation

Chapter 33: ML System Design

- Model serving architectures
- Feature stores
- A/B testing infrastructure
- Model monitoring and retraining
- Batch vs real-time inference

Part 10: Mastery

Chapter 34: Trade-offs and Decision Making

- When to use what technology
- Cost considerations
- Team expertise and maintainability
- Time-to-market vs perfection
- Technical debt management

Chapter 35: Evolution and Migration

- Zero-downtime deployments
- Blue-green deployments
- Canary releases
- Database migrations at scale
- Backward compatibility

Chapter 36: Advanced Case Studies

- Netflix architecture
- Facebook/Meta architecture
- Amazon architecture
- Google's infrastructure
- Lessons from large-scale failures

Learning Tips

1. **Start with fundamentals** - Don't skip the basics, they're crucial for understanding advanced concepts
2. **Practice calculations** - Learn to estimate capacity, storage, and bandwidth requirements
3. **Draw diagrams** - Visualize every system you learn about
4. **Study real systems** - Read engineering blogs from major tech companies
5. **Practice mock interviews** - Use platforms like Pramp or interviewing.io
6. **Build projects** - Implement simplified versions of systems you study
7. **Iterate and improve** - First make it work, then make it scalable, then make it efficient

Recommended Study Path

- **Weeks 1-2**: Chapters 1-3 (Foundation)
- **Weeks 3-5**: Chapters 4-8 (Core Building Blocks)
- **Weeks 6-7**: Chapters 9-11 (Communication)
- **Weeks 8-9**: Chapters 12-14 (Reliability)
- **Weeks 10-13**: Chapters 15-20 (Advanced Concepts)
- **Weeks 14-15**: Chapters 21-25 (Specialized Topics)
- **Weeks 16-17**: Chapters 26-28 (Patterns & Optimization)
- **Weeks 18-20**: Chapter 29-30 (Practical Design)
- **Weeks 21-22**: Chapters 31-33 (Modern Architecture)
- **Weeks 23-24**: Chapters 34-36 (Mastery & Review)