SWE/Backend Interview Prep Checklist

OPERATING SYSTEMS

□ Processes & Threads

- □ Difference between process and thread
- □ Context switching
- □ Process states (new, ready, running, waiting, terminated)
- □ Multithreading vs multiprocessing
- □ Thread synchronization basics

□ Memory Management

- □ Virtual memory concept
- □ Paging and segmentation (high level)
- □ Page replacement algorithms
  - □ LRU, FIFO
- □ Stack vs Heap memory
- □ Memory leaks and prevention

□ Concurrency

- □ Race conditions
- □ Deadlocks
  - □ Necessary conditions
  - □ Prevention
- □ Mutex vs Semaphore
- □ Critical section problem
- □ Producer-consumer problem

□ CPU Scheduling

- □ FCFS, Round Robin, Priority scheduling
- □ Context switching overhead
- □ Preemptive vs non-preemptive

□ File Systems & IPC

- □ Inodes, file permissions
- □ Hard vs soft links
- □ Pipes, shared memory, message queues, sockets

COMPUTER NETWORKS

□ OSI Model & TCP/IP

- □ 7 layers of OSI
- □ TCP/IP 4-layer model
- □ Protocols at each layer

□ HTTP/HTTPS

- □ Methods: GET, POST, PUT, DELETE, PATCH
- □ Status codes (200, 201, 400, 401, 403, 404, 500)
- □ Headers (Authorization, Content-Type, Cache-Control)
- □ HTTP vs HTTPS, HTTP/1.1 vs HTTP/2
- □ Cookies, sessions, tokens

□ TCP vs UDP

- □ Differences and use cases
- □ Three-way handshake
- □ ACK and retransmission

□ DNS & Real-Time

- □ DNS resolution process
- □ DNS record types (A, CNAME, MX)
- □ WebSockets and Server Sent Events

□ RESTful APIs

- □ REST principles and statelessness
- □ Resource-based URLs and Idempotency

□ Security Basics

- □ SSL/TLS basics
- □ JWT tokens
- □ OAuth 2.0 flow
- □ CORS, CSRF, XSS

SYSTEM DESIGN

□ Scalability Basics

- □ Horizontal vs vertical scaling
- □ Load balancing (round robin, least connections)
- □ Stateless vs stateful services

□ Caching

- □ Cache locations (client, CDN, server, DB)
- □ Invalidation strategies
- □ Eviction policies (LRU, LFU)
- □ Redis and Memcached basics

□ Databases at Scale

- □ Replication (master-slave, master-master)
- □ Sharding/partitioning
- □ Connection pooling

□ APIs & Messaging

- □ REST vs GraphQL
- □ Rate limiting and API versioning
- □ RabbitMQ and Kafka basics

□ Architecture Patterns

- □ Monolith vs Microservices
- □ Serverless, Service Mesh
- □ Event Sourcing, CQRS

DATABASE MANAGEMENT SYSTEMS

□ **SQL Fundamentals**

- □ SELECT, JOINs (inner, left, right, full)
- □ GROUP BY, HAVING, aggregate functions
- □ Subqueries, nested queries
- □ Window functions (ROW_NUMBER, RANK, LAG, LEAD)
- □ UNION vs UNION ALL

□ **Database Design**

- □ Normalization (1NF, 2NF, 3NF)
- □ When to denormalize
- □ Primary, foreign, composite keys
- □ ER diagrams (basic understanding)

□ **Indexing**

- □ What indexes are, how they work
- □ B-tree vs Hash index
- □ Clustered vs non-clustered index
- □ Composite indexes
- □ When indexes help vs hurt

□ **Transactions & ACID**

- □ Atomicity, Consistency, Isolation, Durability
- □ Isolation levels (4 types)
- □ Dirty reads, phantom reads, non-repeatable reads
- □ Commit, rollback, savepoints

□ **Concurrency Control & Scaling**

- □ Locking mechanisms (shared, exclusive)
- □ Optimistic vs pessimistic locking
- □ Replication (master-slave, master-master)
- □ Sharding/partitioning strategies
- □ N+1 query problem

□ **SQL vs NoSQL**

- □ When to use which
- □ CAP theorem (basic understanding)
- □ NoSQL types: Document, Key-Value, Column, Graph

BACKEND-SPECIFIC TECHNOLOGIES

□ **Python Backend**
- □ Frameworks: FastAPI or Django
- □ Async programming (asyncio, async/await)
- □ ORM: SQLAlchemy or Django ORM
- □ Testing: pytest, unittest

□ **REST API Design**
- □ Resource naming conventions
- □ HTTP methods and status codes
- □ Swagger/OpenAPI documentation

□ **Authentication & Authorization**
- □ JWT tokens
- □ OAuth flow
- □ Session-based auth

ESSENTIAL TOOLS

□ **Git**
- □ Branching, merging, rebasing, conflict resolution

□ **Docker**
- □ Containers, Dockerfile, docker-compose
- □ Containerization vs Virtualization

□ **CI/CD & Testing**
- □ Basic pipeline understanding
- □ Test Driven Development

□ **Linux/Bash**
- □ Basic commands, file permissions, process management

□ **Web Servers**
- □ Nginx, Apache, Caddy, MS IIS

API TYPES & DATABASES

□ **API Styles**
- □ REST (HATEOAS, JSON APIs)
- □ GraphQL
- □ gRPC and protocol buffers
- □ SOAP basics

□ **SQL Databases**
- □ PostgreSQL, MySQL, MS SQL, Oracle, MariaDB

□ **NoSQL Databases**
- □ Key-Value: Redis, DynamoDB
- □ Document: MongoDB, CouchDB
- □ Graph: Neo4j
- □ Time Series: InfluxDB, TimeScale
- □ Column: Cassandra, HBase
- □ Real-time: Firebase, RethinkDB

□ **Search & Tools**
- □ Elasticsearch, Solr

DATABASE SCALING & PERFORMANCE

□ **Sharding Strategies**
- □ Horizontal sharding
- □ Hash-based sharding
- □ Range-based sharding

□ **Data Replication**
- □ Master-slave replication
- □ Master-master replication
- □ Read replicas

□ **Query Performance**
- □ Database indexes and index strategies
- □ N+1 query problem
- □ Query optimization

□ **Other Concepts**
- □ CAP theorem
- □ Connection pooling

RELIABILITY & RESILIENCE

□ **Resilience Patterns**
- □ Circuit Breaker
- □ Graceful Degradation
- □ Throttling, Backpressure
- □ Loadshifting, Bulkhead Pattern

□ **Migration & Deployment**
- □ Database migrations
- □ API versioning migrations
- □ Zero-downtime deployments
- □ Rollback strategies

□ **Types of Scaling**
- □ Database, application, infrastructure, geographic

SECURITY

□ **Hashing & Encryption**
- □ MD5, SHA, bcrypt, scrypt, Argon2
- □ SSL/TLS, HTTPS, end-to-end encryption

□ **Web Security**
- □ OWASP Top 10
- □ CSRF, XSS, SQL injection prevention
- □ CSP (Content Security Policy)

□ **API & Server Security**
- □ Rate limiting, input validation
- □ Output encoding, API key management
- □ Secure headers

OBSERVABILITY & MONITORING

□ **Metrics, Logging, Tracing**

- □ Instrumentation
- □ Monitoring (application, infrastructure)
- □ Telemetry collection
- □ Distributed tracing, log aggregation

□ **Monitoring Tools**

- □ Prometheus, Grafana
- □ ELK Stack, Jaeger
- □ DataDog, New Relic

MESSAGE BROKERS & REAL-TIME

□ **Message Brokers**

- □ RabbitMQ, Kafka
- □ Redis Pub/Sub
- □ Async processing, decoupling, event streaming

□ **Real-Time Patterns**

- □ Long Polling, Short Polling
- □ WebSockets, Server Sent Events (SSE)
- □ Pub/Sub patterns

□ **Real-Time Databases**

- □ Firebase Realtime, RethinkDB
- □ Redis Pub/Sub, Kafka streaming

INTERNET FUNDAMENTALS

□ **How Internet Works**

- □ Internet architecture, network layers
- □ Data transmission, bandwidth, latency

□ **Domains & Hosting**

- □ DNS system, domain registration
- □ Hosting providers, CDN basics

□ **Browsers**

- □ How browsers work, client-side rendering
- □ Caching mechanisms

OOP & DESIGN PATTERNS

□ **4 Pillars of OOP**

- □ Encapsulation, Abstraction
- □ Inheritance, Polymorphism

□ **SOLID Principles**

- □ Single Responsibility, Open/Closed
- □ Liskov Substitution, Interface Segregation
- □ Dependency Inversion

□ **Design Patterns**

- □ Creational: Singleton, Factory
- □ Structural: Decorator, Adapter
- □ Behavioral: Observer, Strategy

ARCHITECTURAL PATTERNS

□ **Application Architecture**

- □ Monolithic Apps, Microservices
- □ Service Mesh, SOA, Serverless

□ **Design Principles**

- □ Twelve Factor Apps
- □ Domain Driven Design
- □ Event Sourcing, CQRS

□ **Infrastructure**

- □ Infrastructure as Code
- □ Container orchestration, Kubernetes
- □ Service discovery, rolling updates

CACHING & PERFORMANCE

□ **Caching Strategies**

- □ Cache invalidation strategies
- □ Eviction policies (LRU, LFU)
- □ Client, server, CDN, edge caching
- □ Cache warming

□ **Building For Scale**

- □ Horizontal and vertical scaling
- □ Load balancing strategies
- □ Stateless vs stateful services

DSA (Data Structures & Algorithms) Checklist

Check off topics as you complete them | 180-Day Preparation Guide | Target: 150-200 problems

CORE DATA STRUCTURES

□ **Arrays & Strings**
- □ Two pointers technique
- □ Sliding window
- □ Prefix sums
- □ Kadane's algorithm

□ **Hash Tables**
- □ Collision handling
- □ When to use hash tables
- □ Time complexity analysis

□ **Linked Lists**
- □ Reversal
- □ Cycle detection
- □ Dummy nodes
- □ Fast/slow pointers

□ **Stacks & Queues**
- □ Monotonic stack
- □ Deque operations

□ **Trees**
- □ Binary trees
- □ Binary Search Trees (BST)
- □ In-order traversal (recursive & iterative)
- □ Pre-order traversal (recursive & iterative)
- □ Post-order traversal (recursive & iterative)
- □ Level-order traversal

□ **Heaps**
- □ Min heap
- □ Max heap
- □ Heapify operation
- □ Top K problems

□ **Graphs**
- □ Adjacency list representation
- □ Adjacency matrix representation
- □ DFS (Depth-First Search)
- □ BFS (Breadth-First Search)
- □ Topological sort

□ **Tries**
- □ Basic implementation
- □ Prefix searching

CORE ALGORITHMS - SORTING & SEARCHING

□ **Sorting Algorithms**
- □ Quicksort
- □ Mergesort
- □ Time complexity analysis
- □ Space complexity analysis

□ **Binary Search**
- □ Binary search on arrays
- □ Binary search on answer space
- □ Rotated arrays

CORE ALGORITHMS - ADVANCED

□ **Recursion & Backtracking**
- □ Subsets
- □ Permutations
- □ Combinations

□ **Dynamic Programming - 1D**
- □ Climbing stairs
- □ House robber
- □ Coin change

□ **Dynamic Programming - 2D**
- □ Longest common subsequence
- □ Edit distance
- □ Knapsack problem

□ **Greedy Algorithms**
- □ Interval problems
- □ Activity selection

□ **Graph Algorithms**
- □ Dijkstra's algorithm (shortest path)
- □ Union-Find (connected components)
- □ BFS for shortest path
- □ DFS for connected components

DO NOT NEED

□ **Advanced Competitive Programming**
- □ Segment trees
- □ Fenwick trees
- □ Heavy math/number theory
- □ Floyd-Warshall algorithm
- □ Bellman-Ford algorithm
- □ Obscure algorithms