Table of Contents

[**🔹 Architecture & Design Patterns** 1](#_Toc190542018)

[**🔹 Scalability & Performance** 2](#_Toc190542019)

[**🔹 Data Management & Storage** 2](#_Toc190542020)

[**🔹 Networking & Communication** 3](#_Toc190542021)

[**🔹 Reliability & Availability** 3](#_Toc190542022)

[**🔹 Security & Authentication** 4](#_Toc190542023)

[🔹 **DevOps & Monitoring** 6](#_Toc190542024)

[**🔹 Distributed Systems & Consensus** 7](#_Toc190542025)

[**🔹 Containerization & Orchestration** 7](#_Toc190542026)

[**🔹 Cloud Computing & Edge Computing** 8](#_Toc190542027)

[**🔹 Fault Tolerance & Resilience** 8](#_Toc190542028)

[**🔹 API Design & Optimization** 8](#_Toc190542029)

[**🔹 Search & Recommendation Systems** 8](#_Toc190542030)

[**🔹 Emerging Trends** 9](#_Toc190542031)

**🔹 Architecture & Design Patterns**

1. **Monolithic Architecture** – A single, tightly coupled application where all components (UI, logic, database) reside in one unit.  
   **Example**: Traditional e-commerce platforms like early Amazon.
2. **Microservices Architecture** – An application is divided into independent services communicating via APIs.  
   **Example**: Netflix's streaming service with separate services for recommendations, billing, and playback.
3. **Service-Oriented Architecture (SOA)** – A collection of loosely coupled services that communicate using a standard protocol.  
   **Example**: Banking systems where different services (loans, accounts, payments) interact over SOAP/REST APIs.
4. **Event-Driven Architecture** – Services communicate asynchronously via events rather than direct API calls.  
   **Example**: Uber's ride-matching system where events trigger updates on driver availability and ride requests.
5. **Layered Architecture** – Organizes software into layers (Presentation, Business Logic, Data Access) for modularity.  
   **Example**: MVC frameworks like Spring Boot and .NET applications.
6. **Hexagonal Architecture (Ports & Adapters)** – Separates business logic from external dependencies like databases.  
   **Example**: Enterprise applications using Domain-Driven Design (DDD).
7. **Serverless Architecture** – Code execution managed by cloud providers, scaling automatically based on demand.  
   **Example**: AWS Lambda for processing image uploads.
8. **CQRS (Command Query Responsibility Segregation)** – Separates read and write operations into different models.  
   **Example**: Financial trading systems where fast reads and write consistency are required.
9. **Strangler Pattern** – Gradual migration from a legacy system to a new one by replacing functionalities incrementally.  
   **Example**: Migrating from a monolithic to microservices-based e-commerce platform.
10. **Saga Pattern** – A mechanism for handling distributed transactions by chaining multiple steps with compensating actions.  
    **Example**: Order processing in an e-commerce system, where failures trigger refund or inventory rollback.

**🔹 Scalability & Performance**

1. **Horizontal Scaling** – Increasing system capacity by adding more servers.  
   **Example**: Google Search distributing traffic across thousands of servers.
2. **Vertical Scaling** – Upgrading a server’s resources (CPU, RAM) to handle more load.  
   **Example**: Running a database on a high-performance AWS EC2 instance.
3. **Load Balancing** – Distributing incoming traffic across multiple servers to prevent overload.  
   **Example**: Using Nginx to balance requests between web servers.
4. **Sharding** – Splitting a database into smaller pieces (shards) based on criteria like user ID or region.  
   **Example**: Instagram storing different user groups on different database shards.
5. **Replication** – Creating multiple copies of data to improve read performance and fault tolerance.  
   **Example**: MySQL Master-Slave replication for read-heavy applications.
6. **Caching** – Storing frequently accessed data in memory to reduce response times.  
   **Example**: Redis caching API responses in a news website.
7. **Indexing** – Optimizing database queries by using data structures like B-Trees or Hash Indexes.  
   **Example**: Creating indexes on email in a user authentication system.
8. **Rate Limiting** – Restricting the number of API requests per second to prevent abuse.  
   **Example**: Twitter limiting API calls to avoid DDoS attacks.
9. **Content Delivery Network (CDN)** – A network of globally distributed servers caching and serving content closer to users.  
   **Example**: Cloudflare caching website images for faster delivery.
10. **Throttling** – Controlling system resource usage to prevent overload.  
    **Example**: A payment API delaying or rejecting excess transactions during peak hours.

**🔹 Data Management & Storage**

1. **SQL (Relational Database)** – Structured data storage using tables with ACID compliance.  
   **Example**: PostgreSQL for banking applications.
2. **NoSQL (Non-Relational Database)** – Schema-less databases designed for scalability and flexibility.  
   **Example**: MongoDB storing user profiles in a social media app.
3. **Database Partitioning** – Dividing a database into smaller, independent partitions to improve efficiency.  
   **Example**: Amazon DynamoDB partitioning user data based on geographic regions.
4. **Event Sourcing** – Storing all changes to data as immutable events rather than just the latest state.  
   **Example**: Financial ledgers tracking transaction history.
5. **Data Lake** – A repository storing raw, unstructured data for analytics.  
   **Example**: AWS S3 storing logs, images, and analytics data for machine learning.
6. **Data Warehouse** – A structured, optimized storage system for business intelligence queries.  
   **Example**: Google BigQuery analyzing large datasets.
7. **Polyglot Persistence** – Using different types of databases (SQL, NoSQL) within the same system.  
   **Example**: A retail website using MySQL for orders and Redis for caching.
8. **Message Queue (MQ)** – Asynchronous messaging between services to improve decoupling.  
   **Example**: RabbitMQ managing email notifications in an e-commerce platform.
9. **Distributed File System** – A file storage system that spans multiple servers.  
   **Example**: Google File System (GFS) for cloud storage.
10. **CAP Theorem** – A principle stating that a distributed system can achieve only two out of **Consistency, Availability, and Partition Tolerance**.  
    **Example**: MongoDB prioritizing Availability and Partition Tolerance.

**🔹 Networking & Communication**

1. **REST API** – A stateless API that follows HTTP conventions for communication.  
   **Example**: OpenWeather API providing weather data.
2. **GraphQL** – A flexible query language for APIs that allows clients to request specific data.  
   **Example**: GitHub API enabling selective data fetching.
3. **gRPC** – A high-performance RPC framework using Protocol Buffers for communication.  
   **Example**: Google Kubernetes Engine (GKE) managing cluster nodes via gRPC.
4. **WebSockets** – A protocol for real-time, bidirectional communication between a client and server.  
   **Example**: WhatsApp’s real-time messaging system.
5. **Reverse Proxy** – A server that forwards client requests to backend servers for security and performance.  
   **Example**: Nginx acting as a reverse proxy for a Node.js application.
6. **API Gateway** – A central entry point for managing API requests, authentication, and rate limiting.  
   **Example**: AWS API Gateway for serverless applications.
7. **Circuit Breaker Pattern** – Prevents cascading failures by stopping calls to a failing service.  
   **Example**: Netflix Hystrix handling service failures gracefully.
8. **Service Discovery** – A mechanism for services to locate and communicate with each other dynamically.  
   **Example**: Kubernetes’ built-in service discovery for microservices.
9. **Sidecar Pattern** – Deploying auxiliary services alongside an application for logging, monitoring, or security.  
   **Example**: Istio’s Envoy proxy managing microservice communication.

**🔹 Reliability & Availability**

1. **Fault Tolerance** - The ability of a system to continue operating correctly even when some of its components fail. Achieved through redundancy, failover, and replication mechanisms. The goal is to ensure **high availability**, **reliability**, and **graceful degradation** rather than a complete system crash.
2. **High Availability (HA)** – Ensuring minimal downtime using redundancy and failover mechanisms.  
   **Example**: Google Cloud’s multi-region database setup.
3. **Failover** – Automatically switching to a backup system when the primary system fails.  
   **Example**: AWS RDS Multi-AZ Failover for databases.
4. **Disaster Recovery** – Strategies to restore services after failures.  
   **Example**: AWS backups for restoring lost data.
5. **Leader Election** – Selecting a primary node to coordinate tasks in a distributed system.  
   **Example**: Apache ZooKeeper managing leader election in Kafka.
6. **Eventual Consistency** – A system where data updates eventually propagate across all nodes.  
   **Example**: Amazon DynamoDB ensuring consistency across regions.
7. **Replication** – Creating multiple copies of data across different servers to improve availability and reliability.  
   **Example**: MySQL master-slave replication for database redundancy.
8. **Redundancy** – Duplicating system components to prevent failures from causing downtime.  
   **Example**: A secondary data center in a different location serving as a backup for disaster recovery.
9. **Quorum-Based Decision Making** – Requiring agreement from a majority of nodes before committing changes in distributed systems.  
   **Example**: Apache ZooKeeper using quorum-based writes to maintain consistency.
10. **Graceful Degradation** – A system's ability to continue functioning at a reduced capacity instead of failing completely.  
    **Example**: A video streaming service lowering video resolution when network bandwidth is low.
11. **Circuit Breaker Pattern** – A design pattern preventing a system from making repeated failed requests to an unresponsive service.  
    **Example**: Netflix’s Hystrix library automatically stopping calls to failing microservices.
12. **Load Balancing** – Distributing network traffic across multiple servers to ensure high availability.  
    **Example**: AWS Elastic Load Balancer (ELB) routing user requests to multiple EC2 instances.
13. **Sharding** – Splitting a database into smaller, distributed partitions to improve performance and availability.  
    **Example**: Twitter sharding user data across multiple databases to handle billions of tweets.
14. **Eventual Consistency** – Ensuring that all replicas of data will eventually become consistent after updates.  
    **Example**: DynamoDB’s eventual consistency model allowing fast writes with delayed synchronization.
15. **Strong Consistency** – Ensuring all nodes see the latest data immediately after an update.  
    **Example**: Google Spanner using TrueTime API for globally consistent transactions.
16. **Leader Election** – Choosing a single node in a distributed system to coordinate tasks.  
    **Example**: Apache ZooKeeper managing leader election in distributed applications.

### **🔹 Security & Authentication**

1. **OAuth** – An open standard for secure access delegation, allowing third-party apps to access user data without exposing passwords.  
   **Example**: Logging into Spotify using Google or Facebook credentials.
2. **JWT (JSON Web Token)** – A self-contained token used for securely transmitting information between parties.  
   **Example**: A React app storing a JWT in local storage for user authentication.
3. **SSO (Single Sign-On)** – A mechanism that enables users to log in once and access multiple applications without re-entering credentials.  
   **Example**: Google Workspace allowing access to Gmail, Google Drive, and YouTube with a single login.
4. **LDAP (Lightweight Directory Access Protocol)** – A protocol for managing and authenticating users in directory-based systems.  
   **Example**: Active Directory using LDAP for centralized enterprise authentication.
5. **RBAC (Role-Based Access Control)** – Restricts user access based on predefined roles.  
   **Example**: A healthcare system where only doctors can view patient records, while nurses can only update vitals.
6. **ABAC (Attribute-Based Access Control)** – Grants access based on user attributes like department, location, or device type.  
   **Example**: A corporate VPN allowing access only to employees using company-issued devices.
7. **Multi-Factor Authentication (MFA)** – Requires users to provide multiple forms of authentication (e.g., password + OTP).  
   **Example**: A banking app requiring both a password and an SMS code for login.
8. **HMAC (Hash-Based Message Authentication Code)** – Ensures message integrity and authenticity using cryptographic hashing.  
   **Example**: API request authentication using HMAC-SHA256 signatures.
9. **TLS (Transport Layer Security)** – Encrypts data in transit between clients and servers for secure communication.  
   **Example**: HTTPS websites using TLS to protect sensitive data like login credentials.
10. **End-to-End Encryption (E2EE)** – Ensures only communicating users can read messages, preventing intermediaries from accessing data.  
    **Example**: WhatsApp using E2EE to secure private chats.
11. **Public Key Infrastructure (PKI)** – A system of cryptographic keys, certificates, and authorities for secure communication.  
    **Example**: SSL/TLS certificates issued by a trusted CA (Certificate Authority) like Let's Encrypt.
12. **Zero Trust Security** – A security model that assumes no system is inherently trusted and requires verification for every access request.  
    **Example**: Google BeyondCorp requiring continuous authentication even for employees inside the corporate network.
13. **API Key Authentication** – A method where a client includes a secret API key in requests to authenticate itself.  
    **Example**: Google Maps API requiring an API key to access location services.
14. **OAuth Scopes** – Define the level of access granted to third-party applications.  
    **Example**: A fitness tracking app requesting access only to step count data from Google Fit.
15. **Security Token Service (STS)** – Issues temporary security credentials for accessing cloud services securely.  
    **Example**: AWS STS generating short-lived access tokens for federated authentication.
16. **Web Application Firewall (WAF)** – Protects web applications from threats like SQL injection and cross-site scripting (XSS).  
    **Example**: Cloudflare WAF blocking malicious requests targeting a website.
17. **Data Masking** – Hides sensitive information in databases to prevent unauthorized access.  
    **Example**: A credit card number stored as XXXX-XXXX-XXXX-1234 in logs.
18. **Tokenization** – Replaces sensitive data with unique tokens to enhance security.  
    **Example**: Payment gateways like Stripe using tokenization for credit card transactions.
19. **Biometric Authentication** – Uses unique biological traits like fingerprints or facial recognition for authentication.  
    **Example**: Apple Face ID unlocking an iPhone securely.

### **🔹 DevOps & Monitoring**

1. **CI/CD (Continuous Integration & Continuous Deployment)** – Automates code integration, testing, and deployment to production.  
   **Example**: GitHub Actions running automated tests and deploying a React app to Vercel.
2. **Infrastructure as Code (IaC)** – Managing infrastructure using code instead of manual processes.  
   **Example**: Terraform defining cloud resources using configuration files.
3. **Configuration Management** – Automating and maintaining software configurations across environments.  
   **Example**: Ansible provisioning and configuring servers in a Kubernetes cluster.
4. **Containerization** – Packaging applications and dependencies into lightweight, portable containers.  
   **Example**: Docker running a Node.js app in a container on any OS.
5. **Orchestration** – Managing and automating container deployment, scaling, and networking.  
   **Example**: Kubernetes automating deployment and scaling of microservices.
6. **Blue-Green Deployment** – Running two identical environments (blue & green) for seamless updates with minimal downtime.  
   **Example**: A banking app deploying a new version to the "green" environment before switching traffic from "blue."
7. **Canary Deployment** – Rolling out updates to a small subset of users before full deployment.  
   **Example**: Netflix testing new UI changes with 5% of users before a global release.
8. **Feature Flags** – Enabling or disabling features without deploying new code.  
   **Example**: Facebook rolling out a new "dark mode" option to select users using feature flags.
9. **Logging & Log Aggregation** – Collecting and centralizing logs for troubleshooting and analytics.  
   **Example**: ELK Stack (Elasticsearch, Logstash, Kibana) aggregating and visualizing logs from microservices.
10. **Monitoring & Observability** – Tracking system health, performance, and failures in real-time.  
    **Example**: Prometheus monitoring CPU usage and alerting engineers when thresholds are exceeded.
11. **Application Performance Monitoring (APM)** – Monitoring and analyzing app performance metrics.  
    **Example**: Datadog tracking API response times and database query performance.
12. **Distributed Tracing** – Tracking requests as they pass through different services in a distributed system.  
    **Example**: Jaeger tracing a request from a web app to backend services and databases.
13. **Alerting & Incident Management** – Notifying teams when issues occur and managing responses.  
    **Example**: PagerDuty alerting an engineer when a production API goes down.
14. **Chaos Engineering** – Testing system resilience by intentionally introducing failures.  
    **Example**: Netflix’s Chaos Monkey randomly shutting down servers to test system recovery.
15. **Auto Scaling** – Dynamically adjusting server resources based on demand.  
    **Example**: AWS Auto Scaling increasing EC2 instances during peak hours.
16. **Service Mesh** – Managing communication between microservices with built-in security and monitoring.  
    **Example**: Istio providing traffic management, security, and observability in Kubernetes.
17. **Reverse Proxy** – Directing client requests to backend services while handling load balancing and security.  
    **Example**: Nginx acting as a reverse proxy for a cluster of web servers.
18. **API Gateway** – A centralized entry point managing API requests, authentication, and rate limiting.  
    **Example**: AWS API Gateway handling API traffic for a serverless application.
19. **Sidecar Pattern** – Running auxiliary services (logging, monitoring) alongside the main application in containers.  
    **Example**: A logging agent collecting logs from microservices in Kubernetes.
20. **SLO (Service Level Objective)** – A measurable goal for system reliability and performance.  
    **Example**: A SaaS provider defining an SLO of 99.9% uptime per month.
21. **SLA (Service Level Agreement)** – A contract defining service expectations between providers and customers.  
    **Example**: An AWS SLA guaranteeing compensation if uptime drops below 99.99%.
22. **SLI (Service Level Indicator)** – A metric measuring performance against an SLO.  
    **Example**: API response time being measured as an indicator of system reliability.
23. **Error Budget** – The acceptable amount of downtime or errors before violating an SLA.  
    **Example**: A company allowing 0.1% request failures per quarter before triggering an incident response.
24. **Immutable Infrastructure** – Deploying new server instances instead of updating existing ones.  
    **Example**: A Kubernetes cluster replacing outdated pods with fresh deployments.
25. **Secret Management** – Storing and managing sensitive information like API keys and passwords securely.  
    **Example**: HashiCorp Vault managing database credentials in a cloud environment.

**🔹 Distributed Systems & Consensus**

1. **Leader-Follower Pattern** – A distributed system design where one node (leader) manages updates while followers replicate data.  
   **Example**: PostgreSQL primary-replica replication.
2. **Gossip Protocol** – A decentralized way of sharing information across nodes in a distributed system.  
   **Example**: Apache Cassandra propagating cluster state updates.
3. **Raft Consensus Algorithm** – A consensus algorithm used for leader election and distributed coordination.  
   **Example**: HashiCorp Consul using Raft for service discovery.
4. **Paxos Algorithm** – A consensus protocol ensuring consistency in distributed systems.  
   **Example**: Google’s Chubby lock service.
5. **Vector Clocks** – A mechanism for tracking event order in distributed systems to resolve conflicts.  
   **Example**: DynamoDB conflict resolution in multi-master setups.
6. **Two-Phase Commit (2PC)** – A protocol ensuring atomic transactions across multiple distributed nodes.  
   **Example**: Banking systems transferring funds between accounts in different databases.

**🔹 Containerization & Orchestration**

1. **Containers** – Lightweight, portable units of software packaging applications and dependencies.  
   **Example**: Docker running microservices in isolated environments.
2. **Kubernetes** – An open-source container orchestration platform for managing deployments.  
   **Example**: Deploying a multi-container application on Google Kubernetes Engine (GKE).
3. **Service Mesh** – A dedicated infrastructure layer managing service-to-service communication.  
   **Example**: Istio securing and monitoring microservices traffic.
4. **Helm Charts** – A package manager for Kubernetes applications, simplifying deployments.  
   **Example**: Deploying a Redis cluster using Helm in Kubernetes.

**🔹 Cloud Computing & Edge Computing**

1. **IaaS (Infrastructure as a Service)** – Cloud providers offer virtualized computing resources.  
   **Example**: AWS EC2 providing virtual machines.
2. **PaaS (Platform as a Service)** – Cloud-based platforms for building and deploying applications.  
   **Example**: Google App Engine running web applications.
3. **SaaS (Software as a Service)** – Cloud-based software accessed via the internet.  
   **Example**: Gmail, Slack, and Salesforce.
4. **Edge Computing** – Processing data closer to the source rather than a central data center.  
   **Example**: Self-driving cars processing sensor data locally.

**🔹 Fault Tolerance & Resilience**

1. **Retry Mechanism** – Automatically reattempting failed requests.  
   **Example**: A mobile banking app retrying a failed payment transaction.
2. **Dead Letter Queue (DLQ)** – A separate queue storing messages that fail to process.  
   **Example**: AWS SQS DLQ capturing failed messages for debugging.
3. **Backpressure** – A mechanism to prevent system overload by controlling request flow.  
   **Example**: Kafka slowing down consumers when message processing is slow.
4. **Idempotency** – Ensuring that repeated operations produce the same result.  
   **Example**: REST API retrying a payment request without double charging.

**🔹 API Design & Optimization**

1. **Rate Limiting** – Controlling the number of API requests a client can make.  
   **Example**: Twitter’s API restricting excessive requests from a single user.
2. **Pagination** – Splitting large datasets into smaller chunks for efficient retrieval.  
   **Example**: Facebook's news feed loading posts in batches.
3. **HATEOAS (Hypermedia as the Engine of Application State)** – Enhancing REST APIs with navigational links.  
   **Example**: GitHub’s API providing links to related resources.
4. **API Versioning** – Managing different versions of an API for backward compatibility.  
   **Example**: Stripe API using /v1/customers and /v2/customers.

**🔹 Search & Recommendation Systems**

1. **Full-Text Search** – A search engine indexing entire text fields for quick retrieval.  
   **Example**: Elasticsearch powering real-time search on e-commerce sites.
2. **Vector Search** – Using machine learning to find similar items based on embeddings.  
   **Example**: Spotify recommending songs based on audio similarity.
3. **Collaborative Filtering** – Suggesting items based on user behavior and preferences.  
   **Example**: Amazon recommending products based on past purchases.

**🔹 Emerging Trends**

1. **Blockchain** – A decentralized ledger for secure transactions.  
   **Example**: Bitcoin’s distributed blockchain network.
2. **Federated Learning** – Training machine learning models across decentralized devices while maintaining privacy.  
   **Example**: Google’s Gboard keyboard learning from user input without sending data to the cloud.
3. **Quantum Computing** – Leveraging quantum mechanics for solving complex computational problems.  
   **Example**: Google’s Sycamore quantum processor solving optimization problems.