Here are **bullet-point notes** for designing a **Load Balancer**, a key system component for distributing traffic across backend servers:

**🧩 1. Problem Statement**

* Evenly distribute incoming client requests across multiple backend servers.
* Improve **reliability**, **throughput**, **redundancy**, and **scalability**.
* Detect and **route around failures**.

**📎 2. Requirements**

**✅ Functional**

* Accept client requests and forward them to backend servers.
* Support **multiple load balancing strategies**.
* Monitor server health and remove failed nodes from rotation.
* Optionally maintain **session stickiness**.

**❌ Non-Functional**

* High availability and fault-tolerance.
* Low latency routing.
* Horizontally scalable.

**🔁 3. Load Balancing Algorithms**

**🔹 Round Robin**

* Cycles through backend servers.
* Simple but ignores load and response time.

**🔹 Least Connections**

* Chooses server with the fewest active connections.
* Better for long-lived requests.

**🔹 IP Hashing**

* Hash client IP to assign a fixed backend (supports stickiness).
* Useful for session persistence.

**🔹 Weighted Round Robin / Least Connections**

* Prioritize servers with more capacity/resources.

**🔹 Consistent Hashing**

* Useful for cache servers or sticky sessions.

**🧠 4. Components**

* **Frontend Listener**: Accepts incoming requests (HTTP/TCP).
* **Health Checker**: Periodically pings backends to detect failure.
* **Router/Dispatcher**: Applies load balancing logic.
* **Monitoring Module**: Tracks latency, failure rate, and throughput.

**🛠 5. Types of Load Balancers**

**🔹 Layer 4 (Transport Level)**

* Operates on TCP/UDP (IP + port).
* Faster, less overhead.
* Example: AWS NLB

**🔹 Layer 7 (Application Level)**

* Understands HTTP, routes based on URL, headers, cookies.
* Enables smarter decisions.
* Example: AWS ALB, NGINX, HAProxy

**🌐 6. Deployment Strategies**

* **DNS Load Balancing**: Simple, limited control.
* **Client-side Load Balancing**: Used in microservices (e.g., Netflix Ribbon).
* **Reverse Proxy Load Balancer**: Centralized control (NGINX, Envoy).
* **Anycast IP + Global Load Balancers** for geo-routing.

**💾 7. Session Persistence (Sticky Sessions)**

* Use cookies or IP hashing to keep client on same backend.
* Required for stateful applications (e.g., shopping carts).

**🧪 8. Health Checks**

* Periodic checks (e.g., /health) on backends.
* Remove unresponsive or slow servers from pool.
* Graceful re-addition after recovery.

**📊 9. Metrics and Monitoring**

* Track:
  + Request count per server
  + Latency
  + Failure rate
  + Server uptime
* Integrate with Prometheus/Grafana for alerts.

**🔐 10. Security Considerations**

* TLS termination at load balancer.
* DDoS protection (rate limiting, IP blocking).
* Authentication and API key filtering.

**📦 11. Caching and Compression**

* Offload static file caching at LB (CDN or proxy layer).
* GZIP compression to reduce payload size.

**⚙️ 12. Real-World Examples**

* **NGINX**, **HAProxy**, **Envoy**, **AWS ELB/ALB/NLB**, **Cloudflare**, **Kong**