SYSTEM DESIGN SELF-AUDIT

Read through each skill below and **honestly** assign yourself a colour. No faking it — be completely truthful with yourself. This is your personal map, not a performance review.

GREEN	I know this deeply. I can explain trade-offs on a whiteboard to a junior dev and debug it in production.
YELLOW	I have worked with this, but if asked "Why?", I'll get stuck. I lack architectural depth here.
RED	Honestly, no clue. I only know the buzzword to put on my resume.

■ **Goal for the next 6 months:** Turn every ■ Red into ■ Yellow, and every ■ Yellow into ■ Green.

1 FOUNDATION & MINDSET

SKILL / CONCEPT	■	■	■
SOLID Principles Do you actually apply them, or just memorize them for interviews?			
Design Patterns — Creational Factory, Builder, Singleton: when to use each and why.			
Design Patterns — Structural Adapter, Decorator, Facade: composing objects cleanly.			
Design Patterns — Behavioral Strategy, Observer: managing algorithms and event flows.			
Concurrency Basics Threads, Locks, race conditions — handling them in real code, not just theory.			

2 SYSTEM ARCHITECTURE & CORE DECISIONS

SKILL / CONCEPT	■	■	■
Monolith vs. Microservices Exact trade-offs and knowing when each is the right call.			
CAP Theorem Practical application of Consistency, Availability, and Partition Tolerance.			
Communication Protocols REST vs. gRPC vs. GraphQL — which one fits which context and why.			
Sync vs. Async Communication Understanding tightly coupled vs. decoupled systems and their failure modes.			
API Gateways & Load Balancing L4 vs. L7 load balancers, reverse proxies, and routing strategies.			

3 DATA & POLYGLOT PERSISTENCE

SKILL / CONCEPT

Relational DBs (PostgreSQL / MySQL)
ACID properties, indexing strategies, sharding, normalization trade-offs.

NoSQL DBs (MongoDB / Cassandra)
Eventual consistency, document vs. wide-column stores, partition keys.

Caching Strategy (Redis / Memcached)
Write-through, Read-through, Cache-aside, invalidation headaches.

Message Brokers (Kafka / RabbitMQ)
Topics, queues, consumer lag, offset management, at-least-once delivery.

Search Engines (Elasticsearch)
How an inverted index works under the hood; query vs. filter context.

4 ARCHITECTURE PATTERNS & RESILIENCE

SKILL / CONCEPT

Event-Driven Architecture
Pub/Sub model, decoupled workflows, event schema evolution.

CQRS & Event Sourcing
Separating read and write models; replaying events for audit and recovery.

Saga Pattern
Distributed transactions, compensating transactions (e.g., Swiggy/Zomato order failure flow).

Resilience Patterns
Circuit Breaker, Retry with backoff, Rate Limiting, Bulkheads — know when each applies.

5 OPERATIONS, DEVOPS & OBSERVABILITY

SKILL / CONCEPT

Containerization (Docker)
Images, layers, container lifecycle, multi-stage builds.

Orchestration (Kubernetes)
Pods, Deployments, Services — conceptual understanding even without deep Ops expertise.

Observability
Distributed Tracing (OpenTelemetry), Centralized Logging, Metrics & Dashboards.

Security Basics
OAuth 2.0, JWT, IAM roles, Encryption at rest and in transit.

■ NEXT STEPS — YOUR ACTION PLAN

■ Audit First	Every skill you marked ■ RED is your current weak zone. Be honest — no one else is grading this.
■ Pick ONE Red This Weekend	Don't try to fix everything at once. Pick just one RED skill and build a small Proof of Concept (PoC).
■ Learn by Doing	Don't just watch YouTube Shorts. Get your hands dirty — write the code, break it, fix it, understand why.
■ 6-Month Target	Red → Yellow in Month 1-2. Yellow → Green by Month 6. Track your progress monthly.
■ Teach to Learn	The fastest way to go from Yellow to Green? Explain it to someone else. Write a blog, make a video, mentor a junior.