STAGE 1 — Core Go (Strong Foundation)

Go Basics

- Variables, types, constants
- Loops, conditions
- Slices, arrays, maps
- Structs, methods
- Interfaces (VERY IMPORTANT)
- Errors (custom errors)

Go Advanced

- Goroutines
- Channels (buffered/unbuffered)
- Mutex, RWMutex
- WaitGroup
- Select statement
- Context package
- Error wrapping
- Panic & recover
- Reflection basics

Go Memory + Performance

- Stack vs Heap
- Escape analysis
- Pointers
- Garbage collector basics
- Benchmarking (testing, bench)

STAGE 2 — Build REST APIs (Industry Standard)

Frameworks

- **Gin** (Most used)
- Fiber (optional)
- Echo

API Building Skills

- Routing
- Request validation
- JSON handling
- Pagination
- Sorting + filtering
- File upload
- Sending emails
- Middlewares (auth, logging, rate limit)
- JWT authentication
- RBAC (Role-based access control)

STAGE 3 — Databases (Backend Mandatory)

SQL

- PostgreSQL (preferred)
- MySQL (optional)
- Joins, indexing, query optimization
- Transactions
- Prepared statements

ORM in Go

- GORM (most common)
- Models
- Auto migrations
- Associations
- Joins
- Soft delete
- Hooks

STAGE 4 — Cache, Queue, Messaging

Redis

- Cache
- Rate limiting
- Pub/Sub
- TTL
- Redis streams

Kafka / RabbitMQ

- Producers
- Consumers
- Partitions
- Offsets
- Consumer group
- Dead-letter queue

STAGE 5 — Clean Architecture + Scalable Code

Must Learn

- Clean Architecture
- Hexagonal architecture
- Repository pattern
- Dependency Injection
- Environment management
- Versioning (v1, v2 APIs
- Logging
- Tracing

STAGE 6 — DevOps + Cloud (Very Important for Jobs)

Docker

- Dockerfile
- Multi-stage builds
- Volumes
- Networks
- Running Go apps in containers

Kubernetes

- Deployments
- Services
- ConfigMaps
- Secrets
- HPA (autoscale)

CI/CD

- GitHub Actions
- GitLab CI
- Jenkins (optional)

Cloud

- AWS (most important) Learn:
- EC2
- S3
- RDS
- ElasticCache
- EKS (Kubernetes)
- CloudWatch
- SNS

STAGE 7 — System Design (Backend Job Mandatory)

- Load Balancing
- Caching
- API Gateway
- Database sharding
- Replication
- Event-driven architecture
- Rate limiting
- CAP theorem
- Consistency models

Golang Specific System Design

- Worker pool
- Connection pooling
- Backpressure
- Goroutine leak prevention
- Context cancellation patterns

MINOR PROJECTS (Easy + Resume booster)

- URL Shortener
- File upload service (S3)
- Simple chat app (WebSocket)
- Rate limiter
- Clean architecture todo app
- Cronjob worker service
- Web scraping tool
- QR code generator API

MAJOR PROJECTS (must build)

1. E-Commerce Backend (FULL FEATURED)

- Endpoints must include:
- Auth (login, OTP, JWT)
- Products
- Categories
- Cart
- Orders
- Payments
- Admin panel
- Inventory
- Coupons
- Delivery status (SSE or WebSocket)

2.Realtime Food Delivery System (Swiggy/Zomato style)

- Features:
- Live order tracking (SSE/WebSocket)
- Driver location updates
- Payment confirmation
- Notification system
- Admin dashboard
- Microservices with message queue

3. DevOps-Ready Microservices Project

- Make 3 services:
- user-service
- order-service
- notification-service
- Connect via Kafka
  Deploy with Docker + Kubernetes
  Use CI/CD
  Log monitoring with Prometheus + Grafana

Interview Preparation

Golang Interview Topics

- Channels deep-dive
- Interface internals
- Memory management
- Goroutine leaks
- Mutex vs RWMutex
- Concurrency patterns
- Worker pool
- Context cancellation
- Deadlocks
- Race conditions

DSA (easy-medium)

- Arrays
- Strings
- Hashmap
- Stack
- Queue
- Tree basics
- Sorting
- Greedy
- Sliding window