MD-04: System Design

Think at Scale - Architecture Reasoning for Real Systems and Interviews

System design interviews are standard at mid-to-senior levels and are the primary differentiator between junior and senior compensation. This is not about memorizing architecture diagrams - it is about learning to frame trade-offs, estimate scale, and present designs with clarity and confidence. In 6 weeks, you will build the architecture reasoning that separates senior engineers from everyone else.

The focus is on thinking in systems, not drawing boxes.

Why This Course?

The Market Reality

Global Context: System design interviews remain the main separator for mid-level and senior backend roles. Teams value engineers who can reason about scale, reliability, latency, data flow, and trade-offs instead of only writing feature-level code.

Nepal Context: Large-scale system design is the skill that separates senior Nepali engineers from mid-level ones. Companies like F1Soft (serving millions of banking transactions), Hamro Patro (Nepal's most-used app), and Daraz (e-commerce at scale in South Asia) all face the distributed systems challenges this course addresses. Nepali engineers pursuing senior remote roles at international companies consistently need system design interview preparation.

Your Opportunity: This course positions you for **senior software engineer, backend architect, and platform engineer roles** - the highest-paying positions in software.

Nepal-Relevant Reality	Opportunity
F1Soft handles millions of banking transactions	Real distributed systems challenges locally
Hamro Patro is Nepal's most-used app	Scale engineering needed in Nepal
Senior remote roles need system design	Interview readiness for top-tier compensation

Course Snapshot

Parameter	Details
Course Code	MD-04
Title	System Design
Duration	1.5 Months (6 Weeks)
Schedule	Monday to Friday (Mon–Fri, 5 Days/Week), 2 Hours/Day
Total Hours	60 Hours of Live Training
Batch Size	Maximum 10 Students
Course Fee	NPR 22,000
Prerequisites	Comfortable with APIs, databases, and general backend development. Saarathi Gate Assessment (diagnostic, no pass/fail) before Day 1.
Self-Study	Minimum 2 hours/day outside class (mandatory)
Outcome	System Design-Ready Software Engineer / Senior Software Engineer

Your Learning Week

Day	Activity
Mon–Fri	2-hour live class session (whiteboard-first, case-study-based)
Mon–Fri	Minimum 2 hours self-study & design practice (mandatory)
Saturday	No classes - flexible self-study, peer collaboration, design work
Sunday	Whole day self-learn time. Classrooms remain fully open for you to come in, study, collaborate with peers, and build projects. (Highly recommended for networking!)

> *Every student MUST spend at least 2 dedicated hours a day on focused design practice beyond the classroom at home. This is non-negotiable for success, it is what separates graduates who get hired from those who don't.*

Week-by-Week Curriculum

Phase 1: Foundations, Scale Thinking & Data Flow (Weeks 1–3, 3 Weeks, 30 Hours)

Week	Focus Area	What You'll Master
Week 1	System Design Fundamentals & Thinking Big	What system design is, real-world system mapping, key vocabulary, distributed systems introduction
Week 2	Data, Caching & Throughput	Sharding, indexing, caching layers, consistency, replication, queues
Week 3	Capacity Planning & Foundational Designs	Load estimates, bottlenecks, trade-off writing, URL shortener and similar patterns

Phase 2: Real-World Cases & Interview Readiness (Weeks 4–6, 3 Weeks, 30 Hours)

Week	Focus Area	What You'll Master
Week 4	Real-World Design Cases I	Social feed, notifications, search, rate limiting, delivery trade-offs
Week 5	Real-World Design Cases II	Messaging, payments, streaming, collaboration, observability, failure modes
Week 6	Portfolio & Career Launch	Final design portfolio, then 3 dedicated days for resume updates, LinkedIn, mock interviews, and communication drills

Skills You'll Gain

Architecture Concepts

Topic	Proficiency Level
Distributed Systems	Fundamentals & Trade-offs
Caching & CDN	Multi-layer Strategy
Database Sharding	Partitioning & Replication
Message Queues	Async Processing
Load Balancing	Traffic Distribution

Topic	Proficiency Level
CAP Theorem	Consistency vs Availability

Design Skills

Skill	Application
Capacity Estimation	Back-of-envelope calculations
Trade-off Analysis	Architectural decision documentation
Whiteboard Communication	Presenting designs under pressure
Failure Mode Analysis	Identifying and mitigating system risks

Topic Depth and Awareness

Section	Guidance
Purpose	This course intentionally separates what you need to master in depth from what you only need to understand with working awareness.
Depth	The architecture reasoning, trade-off writing, and communication workflows practiced repeatedly in class The execution areas you are expected to perform independently in system-design interviews and reviews The estimation and decision habits most likely to matter in senior engineering conversations
Awareness	Adjacent tools, optional stretch topics, and industry context introduced for broader understanding Concepts you should be able to explain, compare, and recognize even if you are not yet executing them independently Advanced directions for later specialization, higher-level tracks, or guided self-study
How to use this syllabus	Spend most of your self-study time strengthening the depth topics first. Treat awareness topics as context builders that help you make better decisions and understand the larger professional landscape.

Project Pool

> *All options below are **intermediate-level final projects**. Each student chooses **one** final project from this pool. Trainers may run smaller guided exercises during the course, but public phase-wise project sections are intentionally removed so the completion standard stays clear and consistent.*

#	Final Project Choice	What You Will Build	Core Stack / Tools
1	Messaging Service Design Packet	Design a scalable messaging system with storage, delivery flow, fan-out choices, and failure handling.	System design docs, message queues, storage trade-offs, consistency
2	Ride-Hailing Matching Architecture	Design a real-time location and matching system with traffic spikes, geospatial queries, and live updates.	Geospatial modeling, WebSockets, caching, service decomposition
3	Streaming Platform Design Review	Create a defendable architecture for media delivery, CDN strategy, caching, and scale trade-offs.	CDN, caching layers, storage, throughput planning
4	Payments Service Design	Design a transaction-safe service with idempotency, ledger integrity, retries, and operational visibility.	Idempotency, event flow, consistency trade-offs, fault tolerance
5	URL Shortener & Analytics Design	Design a compact service that handles redirects, analytics, abuse controls, and regional read performance.	Hashing, data modeling, caching, analytics design

Career Paths & Trajectory

Role Path	Focus and Proof	Stage and Timeline	What Actually Matters
Software Engineer II / Backend Engineer	Understand and reason about existing architectures instead of treating systems as black boxes. Proof you leave with: System design vocabulary and better codebase understanding	Growth stage - 1–3 years	Understand why current systems work before trying to redesign them in meetings full of confidence and zero evidence.

Role Path	Focus and Proof	Stage and Timeline	What Actually Matters
Senior Software Engineer	Design components and services with clearer trade-off thinking, documentation, and ownership of scaling concerns. Proof you leave with: Design documents, technical ownership, and better decision defense	Senior stage - 3–6 years	Write design docs, lead technical discussions, and defend trade-offs with evidence instead of buzzwords.
Staff Engineer / Technical Lead	Set technical standards and guide multi-service architecture decisions across a broader product area. Proof you leave with: Architecture communication and broader technical influence	Leadership stage - 6+ years	Shape system boundaries, mentor others, and keep complexity proportional to the real problem.
Solutions Architect / Engineering Manager	Align system design decisions with product, cost, team execution, and long-term platform direction. Proof you leave with: Strategic design frameworks and stronger cross-functional reasoning	Alternative senior path - 7+ years	Balance technical direction with business constraints, delivery risk, and organizational scale.

Saarathi Gate & Completion Review

Before You Start: Saarathi Gate Assessment

All students complete the **Saarathi Gate Assessment** before Day 1. It is a short diagnostic review of aptitude, learning behaviour, and thinking style. It has **no pass/fail** and is used only to tailor support from the start.

After Course Completion: Saarathi Completion Review

The **Saarathi Academy Certificate** is issued after the selected final project is completed, documented, and reviewed by the trainer. There is **no separate certification exam** for this course.

Completion Requirements:

1. **Attendance:** Minimum 80% attendance
2. **Weekly Work:** Core deliverables, revision work, and practice tasks completed
3. **Final Project:** One intermediate-level project selected from the project pool and completed to trainer-approved quality

4. **Portfolio Proof:** Screenshots, documentation, case-study notes, or equivalent proof assets updated
5. **Trainer Review:** Practical execution, consistency, communication, and overall growth signed off by the trainer

Enrollment & Next Steps

Next Batch: Starting soon (contact for exact dates) **Offline Location:** Old Baneshwor Chowk, Kathmandu, Nepal **Mode:** Online + Offline **Contact (Call/WhatsApp):** 9761095364, 9744442469

>> [ENROLL NOW] - Limited to 10 seats per batch

Architecture is judgment. In 6 weeks, you will have the reasoning and the portfolio to prove it.

Last Updated: Mar 30, 2026