MD-02: DSA & Coding Interview Prep

Crack the Code - Systematic Problem-Solving for Technical Interviews

Over 80% of tech hiring pipelines use structured DSA assessments. This is not about memorizing solutions - it is about building the pattern recognition, time/space reasoning, and communication fluency that interviewers actually test for. In 8 weeks, you will build a practical problem-solving toolkit and interview confidence that gets you past screening rounds.

The focus is on thinking clearly under pressure, not collecting LeetCode badges.

Why This Course?

The Market Reality

Global Context: Structured coding interviews remain a standard filter across software hiring, especially for engineering roles that value problem-solving under pressure. Teams use DSA rounds to judge reasoning, communication, and trade-off thinking before deeper technical interviews.

Nepal Context: Nepal's IT job market is expanding quickly: over 3,000 software companies operate in the Kathmandu Valley alone (CAN Federation, 2024), and companies like F1Soft, Leapfrog, and Fusemachines routinely test candidates with LeetCode-style coding interviews.

Your Opportunity: This course positions you for **software engineer, backend developer, and platform engineer roles** - every coding interview starts with DSA.

Nepal-Relevant Reality	Opportunity
3,000+ software companies in Kathmandu Valley	Massive local hiring demand
F1Soft, Leapfrog, Fusemachines use DSA interviews	Direct prep for Nepal's top employers
Same interview standard as Bangalore/SF	Level playing field with global candidates

Course Snapshot

Parameter	Details
Course Code	MD-02

Parameter	Details
Title	DSA & Coding Interview Prep
Duration	2 Months (8 Weeks)
Schedule	Monday to Friday (Mon–Fri, 5 Days/Week), 2 Hours/Day
Total Hours	80 Hours of Live Training
Batch Size	Maximum 10 Students
Course Fee	NPR 22,000
Prerequisites	Comfortable with basic programming syntax in at least one language (JavaScript or Python preferred). Saarathi Gate Assessment (diagnostic, no pass/fail) before Day 1.
Self-Study	Minimum 2 hours/day outside class (mandatory)
Outcome	Interview-Ready Software Engineer / Junior Software Engineer

Your Learning Week

Day	Activity
Mon–Fri	2-hour live class session (hands-on, problem-solving)
Mon–Fri	Minimum 2 hours self-study & coding practice (mandatory)
Saturday	No classes - flexible self-study, peer collaboration, problem practice
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 coding 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: Core Data Structures & Problem-Solving Habits (Weeks 1–3, 3 Weeks, 30 Hours)

Week	Focus Area	What You'll Master
Week 1	Mindset Building & Algorithmic Thinking	What algorithms and data structures are, pseudocode, problem-solving frameworks, Big-O introduction
Week 2	Strings and HashMaps	String manipulation, hash-based lookups, frequency counting, sliding window basics
Week 3	Linked Lists and Stacks	Pointer manipulation, fast/slow pointers, monotonic stack, queue patterns

Phase 2: Trees, Graphs & Traversal Patterns (Weeks 4–5, 2 Weeks, 20 Hours)

Week	Focus Area	What You'll Master
Week 4	Binary Trees	DFS, BFS, recursion patterns, BST operations, tree construction
Week 5	Graph Algorithms	Graph DFS/BFS, topological sort, Union Find, shortest path basics

Phase 3: Dynamic Programming & Advanced Patterns (Weeks 6–7, 2 Weeks, 20 Hours)

Week	Focus Area	What You'll Master
Week 6	Dynamic Programming	DP foundations, memoization, tabulation, 1D and 2D DP, common patterns
Week 7	Greedy, Backtracking & Binary Search	Greedy selection, backtracking template, binary search on answer, two-pointer refinement

Phase 4: Mock Interviews & Career Launch (Week 8, 1 Week, 10 Hours)

Week	Focus Area	What You'll Master
Week 8	Interview Mastery & Career Prep	Resume updates, LinkedIn polish, full mock coding interviews, behavioral prep, communication drills

Skills You'll Gain

Data Structures & Algorithms

Topic	Proficiency Level
Arrays, Strings, HashMaps	Pattern Recognition
Linked Lists, Stacks, Queues	Pointer Manipulation
Binary Trees, BSTs	Recursive Traversal
Graphs	DFS/BFS/Topological Sort
Dynamic Programming	State Design & Optimization
Greedy & Backtracking	Selection & Enumeration

Interview Skills

Skill	Application
Problem Decomposition	Breaking complex problems into steps
Time/Space Analysis	Big-O reasoning and trade-offs
Whiteboard Communication	Explaining approach while coding
Edge Case Handling	Identifying and addressing corner cases

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 core problem-solving patterns, communication habits, and interview workflows practiced repeatedly in class The execution areas you are expected to perform independently in timed coding interviews The reasoning habits most likely to matter in screening rounds and mock interviews

Section	Guidance
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	Pattern Portfolio	Build a clean portfolio of solved array, string, hash map, sliding-window, and two-pointer problems with reasoning notes.	LeetCode, Python / Java, pattern notes, complexity analysis
2	Data Structures Mastery Pack	Document stack, queue, linked-list, tree, and heap problems with template solutions and edge-case notes.	LeetCode, GitHub, data structures, problem templates
3	Graph & Dynamic Programming Set	Complete an intermediate graph and DP pack with recurrence reasoning, trade-offs, and mistake logs.	Graphs, DFS / BFS, dynamic programming, solution journal
4	Mock Interview Sprint	Run timed mock interviews, collect feedback, and tighten communication under pressure across multiple problem types.	Mock interviews, timed sessions, feedback log, interview communication
5	Interview Readiness Knowledge Base	Create a personal interview system with patterns, common pitfalls, follow-up questions, and revision loops.	GitHub wiki, revision tracker, pattern library, interview prep system

Career Paths & Trajectory

Role Path	Focus and Proof	Stage and Timeline	What Actually Matters
Interview-Ready Junior Software Engineer	Clear coding rounds with better reasoning, stronger communication, and cleaner problem-solving structure. Proof you leave with: Solved-problem portfolio, interview practice, and stronger coding reasoning	Entry stage - first 0–12 months	Your reasoning process matters more than a perfect first attempt. Explain clearly and handle edge cases honestly.
Software Engineer	Carry DSA habits into production work, code reviews, and feature delivery with cleaner logic and debugging discipline. Proof you leave with: Production-style problem solving and clearer technical communication	Growth stage - 1–3 years	Reliable implementation, better code review learning, and choosing reasonable solutions instead of clever chaos.
Mid-Level Software Engineer / Backend Engineer	Use algorithmic judgment to reason about performance, trade-offs, and maintainable implementation choices. Proof you leave with: System-thinking foundation and deeper review discipline	Specialist stage - 2–5 years	Move from solving problems alone to explaining trade-offs, reviewing code, and improving performance thoughtfully.
Senior Software Engineer / Technical Interviewer	Apply the same clarity in hiring loops, mentoring, and design discussions across teams. Proof you leave with: Hiring insight, technical guidance, and stronger engineering judgment	Senior stage - 5+ years	Use technical judgment across delivery, interviews, and architecture conversations - not just whiteboard practice.

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

Interviews reward preparation, not talent. In 8 weeks, you'll have the patterns and the practice to prove it.

Last Updated: Mar 30, 2026