



12-WEEK LEARNING PLAN

Interview Roadmap

A practical route from first diagnostic to interview day

Four focused sessions each week. Learn a concept, practise it, inspect the attempt, and repair the weakest part. The plan connects coding patterns, system design, behavioral evidence, and company preparation.

THE 12-WEEK ARC



PLANNING ASSUMPTION

7-9 focused hours per week. Targets are practice goals, not a promise of interview outcomes. Adjust pace using evidence from repaired attempts.

Preserves the existing Vault sequence and named learning resources.

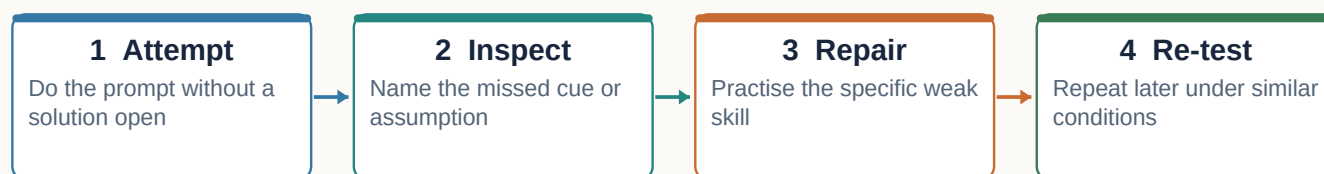
Use evidence to set your pace

The roadmap works best as a weekly feedback loop, not a checklist to finish at any cost.

Who this track is for

Use this plan when you can protect about 7-9 focused hours each week, across four sessions. It fits an interview window that is at least eight weeks away or still flexible.

THE WEEKLY EVIDENCE LOOP



Operating rule

Use the diagnostic to move effort between DSA, system design, and behavioral work. Do not add hours blindly. A repair should name the evidence that would show improvement.

SCORE THE SAME SIX DIMENSIONS

- | | |
|---|--|
| <ul style="list-style-type: none"> ● Problem recognition
0 blocked / 1 inconsistent / 2 reliable | <ul style="list-style-type: none"> ● Correctness and edge cases
0 blocked / 1 inconsistent / 2 reliable |
| <ul style="list-style-type: none"> ● Complexity explanation
0 blocked / 1 inconsistent / 2 reliable | <ul style="list-style-type: none"> ● System-design structure
0 blocked / 1 inconsistent / 2 reliable |
| <ul style="list-style-type: none"> ● Behavioral evidence
0 blocked / 1 inconsistent / 2 reliable | <ul style="list-style-type: none"> ● Communication under time pressure
0 blocked / 1 inconsistent / 2 reliable |

Product note

Vault is the learning foundation. Second Seat is a separately accessed AI-practice layer. Human Calibration is a separately qualified coaching layer. This roadmap embeds no prices; check current product pages for access terms.

Baseline and operating system

Establish a repeatable evidence loop before adding study volume.

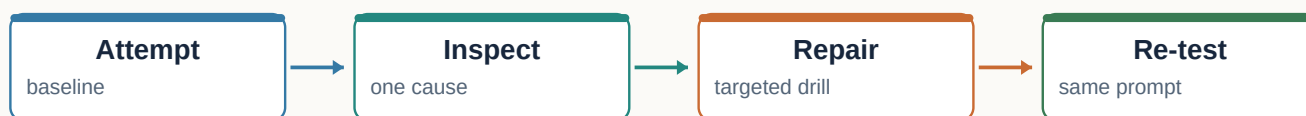
Weekly outcome

Set a realistic weekly rhythm and identify the two skills that need the most attention.

CORE IDEA

A diagnostic is a measurement, not a verdict. Record what happened, the assumption that failed, and the next repair so later attempts are comparable.

HOW IT WORKS



Keep the conditions comparable so improvement is visible.

Real-world application

Real-world application: trace one food-delivery order from menu selection to confirmation. Note which requirements you assumed and what the customer must see if payment is slow.

PRACTICE TARGETS

- Run one mixed DSA diagnostic and one 30-minute system-design diagnostic.
- Start an error log: cue, wrong assumption, correction, and revisit date.
- Draft six raw career stories without polishing them.

VAULT ROUTE

PDF

- operations_study-planning.pdf
- operations_error-log.pdf

TEMPLATES

- template_weekly-prep-tracker.docx
- template_behavioral-story-bank.docx

CHECKPOINT SELF-ASSESSMENT

- Can you name your weakest two dimensions?
- Does each session produce an observable artifact?
- Are four focused sessions protected for next week?

DECISION RULE

Allocate half of Week 2 to the weaker of coding fluency and pattern recognition.



Hash maps, prefix sums, and two pointers

Build pattern recognition and explain the invariant before implementation.

Weekly outcome

Choose among three patterns by recognizing the information the problem needs to retain.

CORE IDEA

Hash maps trade memory for fast keyed lookup. Prefix sums store cumulative totals so a range can be computed by subtraction. Two pointers move boundaries through ordered data while preserving an invariant.

HOW IT WORKS

Pair lookup	→	hash map	Need a fast complement lookup
Range total	→	prefix sum	Subtract two accumulated totals
Sorted pair	→	two pointers	Move a boundary from each end

Real-world application

Real-world application: find whether a basket contains a pair that reaches a target, calculate spending across a run of days, or compare two sorted offer lists. The data shape determines the pattern.

PRACTICE TARGETS

- Solve 15 problems across the three patterns.
- Re-solve four misses after a 48-hour gap.
- Explain the invariant before every second solution.

VAULT ROUTE

VIDEOS

- dsa_hash-maps_01.mp4
- dsa_prefix-sum_01.mp4
- dsa_two-pointers_01.mp4

PDF

- dsa_hash-maps.pdf
- dsa_prefix-sum.pdf
- dsa_two-pointers.pdf

CHECKPOINT SELF-ASSESSMENT

- Can you distinguish keyed lookup, cumulative range, and sorted-boundary work?
- Is repeat accuracy at least 75 percent?

DECISION RULE

Repeat only the failed pattern ladder; advance the others.



Sliding window and binary search

Control boundaries, answer timing, and test selection under time pressure.

Weekly outcome

State what each boundary means and which condition permits it to move.

CORE IDEA

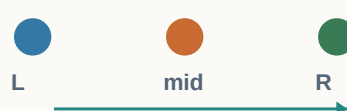
A sliding window maintains a valid contiguous range as its endpoints move. Binary search eliminates a half of an ordered or monotonic search space at every step. Both need precise invariants and termination conditions.

SLIDING WINDOW



Expand or shrink while maintaining the window invariant.

BINARY SEARCH



Discard half only when monotonic order is proven.

Real-world application

Real-world application: find the busiest contiguous 15-minute order window, or locate the first delivery-fee threshold that satisfies a budget. Binary search is valid only when the predicate changes monotonically.

PRACTICE TARGETS

- Solve 14 problems: seven sliding-window and seven binary-search prompts.
- Complete one 45-minute DSA mock with the roadmap rubric.
- Review off-by-one errors as one cluster.

VAULT ROUTE

VIDEOS

- dsa_sliding-window_01.mp4
- dsa_binary-search_01.mp4

PDF

- dsa_sliding-window.pdf
- dsa_binary-search.pdf
- operations_dsa-mock-rubric.pdf

CHECKPOINT SELF-ASSESSMENT

- Can you state the search bounds and stopping rule before coding?
- Can you place a variable-window answer update correctly?

DECISION RULE

A mock below threshold is diagnostic data; repair the failed segment before booking another.

Trees and graph traversal

Choose BFS or DFS from the guarantee you need and make state ownership explicit.

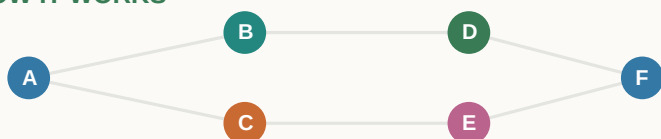
Weekly outcome

Represent the problem as nodes, edges, and visited state before choosing a traversal.

CORE IDEA

BFS explores by distance layers and finds shortest paths in an unweighted graph. DFS follows a path using recursion or an explicit stack and is useful for reachability, components, and structural exploration. Marking visited state prevents repeated work.

HOW IT WORKS



BFS Queue explores by distance / layer.

DFS Stack follows a path before backtracking.

Real-world application

Real-world application: model intersections as vertices and roads as edges. BFS finds the fewest road segments when every edge has equal cost. If travel times differ, use a weighted shortest-path method instead.

PRACTICE TARGETS

- Solve 16 traversal problems across trees, grids, and graphs.
- Write one iterative and one recursive traversal.
- Explain the queue or stack space in terms of width or depth.

VAULT ROUTE

VIDEOS

- dsa_dfs_01.mp4
- dsa_bfs_01.mp4

PDF

- dsa_dfs.pdf
- dsa_bfs.pdf
- dsa_topological-sort.pdf

CHECKPOINT SELF-ASSESSMENT

- Can you define visited-state ownership?
- Can you explain why BFS does or does not give the needed guarantee?

DECISION RULE

If graph setup causes more errors than traversal, isolate representation and visited-state drills.

Backtracking, heaps, intervals, and ordering

Make state restoration and ordering decisions visible before implementation.

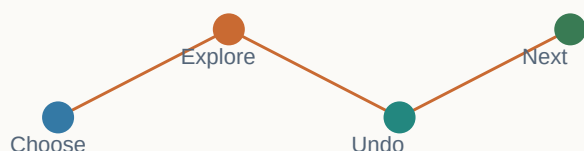
Weekly outcome

Select the smallest tool that matches the structure: branching choices, priority, overlap, or dependencies.

CORE IDEA

Backtracking explores choices and restores state on return. A heap exposes the next highest- or lowest-priority item. Interval algorithms reason about overlap and ordering. Topological sorting orders a directed acyclic dependency graph.

BACKTRACK



ORDERING

earliest

next

later

A heap exposes the next highest-priority item.

Real-world application

Real-world application: build a valid meal bundle by exploring and undoing choices, dispatch the next most urgent order with a heap, and sequence kitchen tasks only after their prerequisites finish.

PRACTICE TARGETS

- Solve 16 problems across the four patterns.
- For backtracking, label choose, explore, and undo.
- For heaps and intervals, state the ordering key before coding.

VAULT ROUTE

VIDEOS

- dsa_backtracking_01.mp4
- dsa_heap-priority-queue_01.mp4
- dsa_intervals_01.mp4
- dsa_topological-sort_01.mp4

PDF

- dsa_backtracking.pdf
- dsa_heap-priority-queue.pdf

CHECKPOINT SELF-ASSESSMENT

- Can you explain why the chosen ordering is enough?
- Do all backtracking branches restore shared state exactly?

DECISION RULE

Keep one weak advanced pattern active next week; park the others after a clean re-solve.

Dynamic programming and midpoint calibration

Derive states and transitions explicitly, then compare evidence with the baseline.

Weekly outcome

Describe a subproblem precisely before writing a recurrence.

CORE IDEA

Dynamic programming stores answers to overlapping subproblems. Define the state, base case, transition, and evaluation order. A one-dimensional state may summarize a sequence prefix; a two-dimensional state may represent two independent positions or capacities.

Write the state, base case, transition, and evaluation order before coding.



State i = best valid answer from earlier states + today's choice

Real-world application

Real-world application: choose a sequence of delivery options with the lowest total cost under a constraint. Each state represents the best result up to a position; the transition compares allowed prior choices. State the constraint before writing the formula.

PRACTICE TARGETS

- Solve 14 DP problems with state, transition, base case, and order written first.
- Run a second DSA mock and compare the same rubric dimensions with Week 3.
- Re-solve the five highest-cost errors from Weeks 1-5.

VAULT ROUTE

VIDEOS

- dsa_dynamic-programming-1d_01.mp4
- dsa_dynamic-programming-2d_01.mp4

PDF

- dsa_dynamic-programming-1d.pdf
- dsa_dynamic-programming-2d.pdf
- operations_dsa-mock-rubric.pdf

CHECKPOINT SELF-ASSESSMENT

- Can you derive a recurrence without copying a template?
- Are repaired mock dimensions improving?

DECISION RULE

Escalate a repeated root cause, not one low score. Human Calibration remains a separate, qualification-based path.



System-design framing and core components

Create the first three traceable portfolio write-ups with one consistent decision structure.

Weekly outcome

Start with the user path and a measurable need; introduce components only to meet that need.

CORE IDEA

A design write-up should connect requirements to estimates, API and data ownership, read/write paths, failure behavior, and an explicit trade-off. A load balancer spreads requests across healthy stateless instances; a cache reduces repeated reads but needs freshness rules.

HOW IT WORKS



Draw the request, response, data owner, and asynchronous boundary.

Real-world application

Real-world application: for a food-delivery menu, route requests across healthy APIs, cache low-change menu data, and send checkout writes to the authoritative order store. Inventory and payment ownership must remain explicit.

PRACTICE TARGETS

- Write #1: load-balanced service, including requirements, estimates, API, data, and bottleneck.
- Write #2: cache-backed read path with invalidation, failure behavior, and one rejected alternative.
- Write #3: API-gateway boundary with routing, authorization, rate protection, and observability.
- Defend one alternative per major component and score one write-up.

VAULT ROUTE

VIDEOS

- sysdesign_load-balancing_01.mp4
- sysdesign_caching_01.mp4
- sysdesign_api-gateway_01.mp4

PDF

- sysdesign_load-balancing.pdf
- sysdesign_caching.pdf
- sysdesign_api-gateway.pdf

CHECKPOINT SELF-ASSESSMENT

- Do you clarify scope before choosing components?
- Can you trace one request end to end?
- Are write-ups #1-#3 independently understandable?

PORTFOLIO CHECKPOINT

Weeks 7-9 build 8 of 10 system-design write-ups. This week completes #1-#3.

DECISION RULE

If structure is weak, repeat the same prompt; if depth is weak, vary the bottleneck.



Data, scale, and failure

Make growth, consistency, and recovery behavior measurable in two new designs.

Weekly outcome

Tie each scaling choice to a measured limit and state the consistency or recovery cost it adds.

CORE IDEA

Partitioning divides data across storage owners but creates routing and hot-key risks. Replication improves read availability but introduces lag and conflict policy. Queues absorb bursts while adding delay, retries, and eventual consistency; dead-letter handling exposes poison messages.

HOW IT WORKS



Measure partition skew and oldest-event age; throttle producers when consumers fall behind.

Real-world application

Real-world application: a restaurant event queue can absorb a dinner rush. Monitor oldest-event age and consumer lag; apply backpressure before the queue grows without bound. Partition by a key that preserves required ordering.

PRACTICE TARGETS

- Write #4: partitioning, consistency, migration, and hot-key handling.
- Write #5: asynchronous processing with bounded retry, dead letters, backpressure, and recovery.
- Estimate traffic, storage, and bandwidth; run one region-down, slow-dependency, or hot-partition drill.

VAULT ROUTE

VIDEOS

- sysdesign_database-replication_01.mp4
- sysdesign_database-sharding_01.mp4
- sysdesign_message-queues_01.mp4
- sysdesign_consistent-hashing_01.mp4

PDF

- sysdesign_database-sharding-tradeoff-matrix.pdf

CHECKPOINT SELF-ASSESSMENT

- Can you explain the consistency consequence of each data choice?
- Can you identify the first scaling limit quantitatively?
- Are write-ups #4-#5 linked to failure drills?

PORTFOLIO CHECKPOINT

Cumulative system-design portfolio: 5 of 10 write-ups (#1-#5).

DECISION RULE

Do not add components unless a stated requirement or measured bottleneck justifies them.



Applied system designs

Add three archetypes and practise changing requirements without losing coherence.

Weekly outcome

Use one clear request path for each archetype, then test how one changed requirement affects it.

CORE IDEA

A URL shortener maps a compact key to a durable destination. Chat systems need conversation ordering and delivery state. Notification systems separate durable intent from provider delivery, retries, and user-visible status.

HOW IT WORKS

- **URL shortener**  key → durable mapping
- **Chat**  ordered conversation events
- **Notifications**  queue → provider → status

Real-world application

Real-world application: a receipt notification may be retried after a provider timeout, but an idempotency key must prevent duplicate customer-visible sends. Contrast that with a chat message, where per-conversation ordering matters.

PRACTICE TARGETS

- Write #6: URL shortener; #7: chat; #8: notifications, each with a 45-minute first pass.
- Answer two follow-up pivots per design without starting over.
- Run one timed design mock and record the weakest rubric dimension.

VAULT ROUTE

VIDEOS

- sysdesign_url-shortener_01.mp4
- sysdesign_chat-system_01.mp4
- sysdesign_notification-system_01.mp4

PDF

- sysdesign_url-shortener-walkthrough.pdf
- sysdesign_chat-system-walkthrough.pdf
- sysdesign_notification-system-walkthrough.pdf

CHECKPOINT SELF-ASSESSMENT

- Can you preserve a coherent design when requirements change?
- Do you explain trade-offs before naming technologies?
- Are write-ups #6-#8 saved with follow-up decisions?

PORTFOLIO CHECKPOINT

Cumulative system-design portfolio: 8 of 10 write-ups (#1-#8).

DECISION RULE

Choose the next company-calibrated design from the weakest applied archetype.

Behavioral evidence and US-context translation

Turn career history into distinct, truthful stories a new listener can calibrate.

Weekly outcome

Make your individual decisions and results understandable without inflating title or scope.

CORE IDEA

The STAR structure gives a listener context (Situation), responsibility (Task), personal action (Action), and evidence of outcome (Result). Use it as a guide, not a script; name team contributions accurately and explain unfamiliar role scope plainly.

HOW IT WORKS



Keep your own action distinct from the team's work.

Real-world application

Real-world application: when describing a production incident, state the service impact, your assigned responsibility, the action you personally took, and what changed afterward. Separate measured result from team-wide outcome.

PRACTICE TARGETS

- Build eight distinct STAR stories mapped to 12 competencies.
- Record six answers and remove setup that does not improve understanding.
- Prepare a 90-second explanation of role, company, and scope.

VAULT ROUTE

VIDEOS

- behavioral_star-method_01.mp4
- behavioral_indian-context-for-us-interviewers_01.mp4

PDF

- behavioral_star-template.pdf
- behavioral_question-bank.pdf
- behavioral_translate-indian-titles.pdf

TEMPLATE

- template_behavioral-story-bank.docx

CHECKPOINT SELF-ASSESSMENT

- Does every story separate your actions from the team's?
- Can a listener calibrate scope without inflated title translation?

PORTFOLIO CHECKPOINT

No new system-design write-up this week. Preserve the verified total of 8 of 10.

DECISION RULE

Replace vague stories with specific evidence; do not add adjectives.



Company calibration and complete interview loops

Finish the ten-write-up portfolio inside two evidence-bounded practice loops.

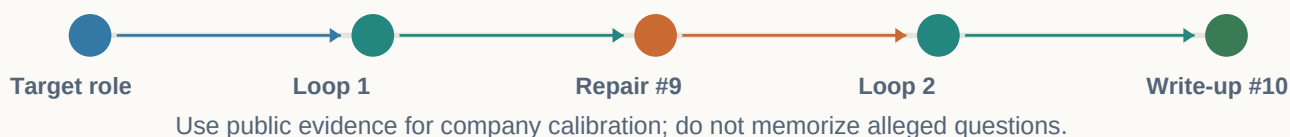
Weekly outcome

Adapt preparation to the role using credible public evidence, not memorized claims about questions.

CORE IDEA

A full loop tests whether skills remain consistent across formats and time pressure. Use the first loop to find a repairable weakness, record the corrected design as write-up #9, then test the repair in a separate loop before writing #10.

HOW IT WORKS



Real-world application

Real-world application: compare two target role descriptions and public engineering material. Select only preparation topics that the evidence supports; write down what remains unknown rather than treating online anecdotes as guarantees.

PRACTICE TARGETS

- Select evidence-approved preparation banks for two target companies.
- Run two full interview loops on separate days.
- Repair the system-design round from each loop as write-ups #9 and #10.
- Repair other misses within 24 hours; repeat only failed segments.

VAULT ROUTE

PDF

- company_master-index.pdf
- operations_interview-day.pdf
- operations_post-interview-debrief.pdf

TEMPLATE

- template_recruiter-screen-notes.docx

CHECKPOINT SELF-ASSESSMENT

- Are two consecutive mock loops near your chosen threshold?
- Can you adapt examples without memorizing alleged questions?
- Do write-ups #9-#10 include decisions, failure path, and repair?

PORTFOLIO CHECKPOINT

System-design portfolio complete: 10 of 10 write-ups (#1-#10).

DECISION RULE

Drop any company bank without adequate public evidence; do not pad it with generic guesses.

Stabilize, apply, and protect interview performance

Consolidate proven behaviors, confirm the portfolio, and reduce avoidable day-of risk.

Weekly outcome

Finish with reliable retrieval and clear logistics, not a last-minute expansion of the syllabus.

CORE IDEA

Spaced retrieval tests whether a skill is available without warm-up. A final full loop checks transitions between interview types. Lowering practice volume before the event preserves recovery time; it does not replace sleep or preparation.

HOW IT WORKS



Reduce volume before the interview; protect sleep, setup, and recovery time.

Real-world application

Real-world application: treat the interview day like a planned trip. Confirm time zone, route or video setup, materials, and buffer time before the event so a preventable logistics failure does not consume attention.

PRACTICE TARGETS

- Re-solve the top 12 errors from a blank editor.
- Run one final full loop, then reduce volume for 48 hours.
- Review all 10 write-ups and rehearse the weakest design's opening, bottleneck, and failure path.
- Confirm time zones, equipment, role scope, and contract-model questions.

VAULT ROUTE

PDF

- behavioral_final-review.pdf
- operations_interview-day.pdf
- remote_contract-red-flags.pdf

TEMPLATES

- template_interview-availability.docx
- template_thank-you.docx

CHECKPOINT SELF-ASSESSMENT

- Is performance stable across coding, design, and behavioral rounds?
- Can you recover aloud when stuck?
- Are all 10 write-ups easy to retrieve?

PORTFOLIO CHECKPOINT

Final verified system-design total: 10 of 10. Week 12 adds no required new write-up.

DECISION RULE

Use the final week to stabilize proven skills, not chase novel hard problems.

Evidence to leave with

Completion means the artifacts exist, can be retrieved, and can be explained. The roadmap does not guarantee an interview outcome.

The evidence portfolio

- **DSA**
75 new problems, spaced re-solves, and a maintained error log.
- **System design**
10 named write-ups: Weeks 7-9 create #1-#8; Week 11 creates #9-#10.
- **Mocks**
At least six timed attempts across interview types, with failed segments repaired.
- **Behavioral**
Eight reusable stories with personal action, evidence, and an honest scope boundary.
- **Company prep**
Two target-company passes grounded in adequate public evidence.

SYSTEM-DESIGN WRITE-UP COUNT



FINAL CONSISTENCY CHECK

- Every checklist item points to an artifact or observable performance result.
- All ten system-design write-ups are numbered and traceable to the named weeks.
- The next action targets the weakest repeated dimension instead of adding random volume.

READINESS GUARDRAIL

Finishing the plan is not itself a readiness signal. Repaired attempts and stable performance provide the evidence. Adjust the next step to the gap that repeats.

REFERENCES AND WORKED PRACTICE

roadmap_12week.pdf

The improved guide ends on the preceding page.

The previous production edition follows in full as an appendix.

It retains its original worked exercises, answers, and references.

Dated standalone edition

https://interviewrunway.com/pdf-guides/practice-2026-10-15/roadmap_12week.pdf

Source references

No source URLs needed restoration in the independent comparison.



12-WEEK LEARNING PLAN

Interview Roadmap

A practical route from first diagnostic to interview day

Four focused sessions each week. Learn a concept, practise it, inspect the attempt, and repair the weakest part. The plan connects coding patterns, system design, behavioral evidence, and company preparation.

THE 12-WEEK ARC



PLANNING ASSUMPTION

7-9 focused hours/week for core. Optional stretch adds time; no interview outcome is promised. Use repaired attempts to adjust pace.

Core pace, eight-week adaptation, and mock cards: pages 16-18. Full-track target: 75 new problems and ten design write-ups.

Learner edition - 9 October 2026.

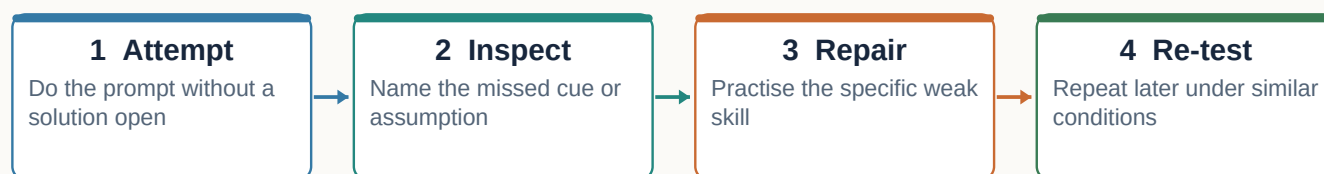
Use evidence to set your pace

The roadmap works best as a weekly feedback loop, not a checklist to finish at any cost.

Who this track is for

Use four protected sessions and 7-9 core hours each week. Allow twelve weeks, or use the conditional eight-week adaptation on page 17. Core and stretch targets are explicit on page 16.

THE WEEKLY EVIDENCE LOOP



Operating rule

Use the diagnostic to move effort between DSA, system design, and behavioral work. Do not add hours blindly. A repair should name the evidence that would show improvement.

DIMENSION CHECKLIST - SCORE BY FORMAT

- | | |
|---|--|
| <ul style="list-style-type: none"> ● Problem recognition
0 blocked / 1 inconsistent / 2 reliable | <ul style="list-style-type: none"> ● Correctness and edge cases
0 blocked / 1 inconsistent / 2 reliable |
| <ul style="list-style-type: none"> ● Complexity explanation
0 blocked / 1 inconsistent / 2 reliable | <ul style="list-style-type: none"> ● System-design structure
0 blocked / 1 inconsistent / 2 reliable |
| <ul style="list-style-type: none"> ● Behavioral evidence
0 blocked / 1 inconsistent / 2 reliable | <ul style="list-style-type: none"> ● Communication under time pressure
0 blocked / 1 inconsistent / 2 reliable |

Product note

Vault is the learning foundation. Second Seat is a separately accessed AI-practice layer. Human Calibration is a separately qualified coaching layer. This roadmap embeds no prices; check current product pages for access terms.

Use page 18 for format anchors and N/A; design uses the active [Timed Technical Rehearsal Workbook](#).

Sign in to your eligible Library account first, then open linked resources.



Baseline and operating system

Establish a repeatable evidence loop before adding study volume.

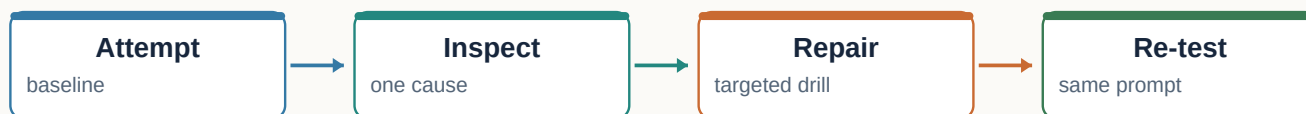
Weekly outcome

Set a realistic weekly rhythm and identify the two skills that need the most attention.

CORE IDEA

A diagnostic is a measurement, not a verdict. Record what happened, the assumption that failed, and the next repair so later attempts are comparable.

HOW IT WORKS



Keep the conditions comparable so improvement is visible.

Real-world application

Real-world application: trace one food-delivery order from menu selection to confirmation. Note which requirements you assumed and what the customer must see if payment is slow.

PRACTICE TARGETS

- Run one mixed DSA diagnostic and one 30-minute system-design diagnostic.
- Start an error log: cue, wrong assumption, correction, and revisit date.
- Draft six raw career stories without polishing them.

CURRENT LIBRARY

PDF - Technical Baseline Scorecard
PDF - Evidence Comparison Correction Workbook
TEMPLATE - Weekly Prep Tracker
TEMPLATE - Behavioral Story Bank

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Can you name your weakest two dimensions?
- Does each session produce an observable artifact?
- Are four focused sessions protected for next week?

DECISION RULE

Allocate half of Week 2 to the weaker of coding fluency and pattern recognition.



Hash maps, prefix sums, and two pointers

Build pattern recognition and explain the invariant before implementation.

Weekly outcome

Choose among three patterns by recognizing the information the problem needs to retain.

CORE IDEA

Hash maps trade memory for fast keyed lookup. Prefix sums store cumulative totals so a range can be computed by subtraction. Two pointers move boundaries through ordered data while preserving an invariant.

HOW IT WORKS

Pair lookup	→	hash map	Need a fast complement lookup
Range total	→	prefix sum	Subtract two accumulated totals
Sorted pair	→	two pointers	Move a boundary from each end

Real-world application

Real-world application: find whether a basket contains a pair that reaches a target, calculate spending across a run of days, or compare two sorted offer lists. The data shape determines the pattern.

CORE + STRETCH PRACTICE

- Core: 9 new problems, three per pattern. Full-track target: 15 with 6 optional stretch problems.
- Re-solve four misses after 48 hours; explain the invariant before each attempt.
- Log independent solves separately from prompted or incomplete attempts.

CURRENT LIBRARY

- [PDF - Hash Maps](#)
- [PDF - Prefix Sum](#)
- [PDF - Two Pointers](#)
- [VIDEO - Two Pointers](#)

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Can you distinguish keyed lookup, cumulative range, and sorted-boundary work?
Author-set target: at least 75% = independent clean re-solves / attempted re-solves. Clean means correct, tested, and explained without a solution open.

DECISION RULE

Repeat only the failed pattern ladder; advance the others.



Sliding window and binary search

Control boundaries, answer timing, and test selection under time pressure.

Weekly outcome

State what each boundary means and which condition permits it to move.

CORE IDEA

A sliding window maintains a valid contiguous range as its endpoints move. Binary search eliminates a half of an ordered or monotonic search space at every step. Both need precise invariants and termination conditions.

HOW IT WORKS

SLIDING WINDOW



Expand or shrink while maintaining the window invariant.

BINARY SEARCH



Discard half only when monotonic order is proven.

Real-world application

Real-world application: find the busiest contiguous 15-minute order window, or locate the first delivery-fee threshold that satisfies a budget. Binary search is valid only when the predicate changes monotonically.

CORE + STRETCH PRACTICE

- Core: 8 new problems, four per pattern. Full-track target: 14 with 6 optional stretch problems.
- Run one 45-minute coding mock; use the format card on page 18.
- Cluster off-by-one errors; repair before the next timed attempt.

CURRENT LIBRARY

- VIDEO** - Sliding Window
- VIDEO** - Binary Search
- PDF** - Sliding Window
- PDF** - Binary Search

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Can you state the search bounds and stopping rule before coding?
- Can you place a variable-window answer update correctly?

DECISION RULE

A mock below threshold is diagnostic data; repair the failed segment before booking another.

Trees and graph traversal

Choose BFS or DFS from the guarantee you need and make state ownership explicit.

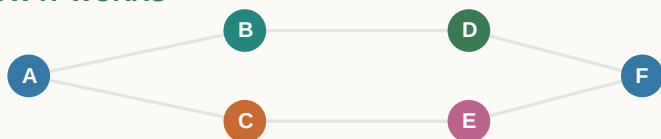
Weekly outcome

Represent the problem as nodes, edges, and visited state before choosing a traversal.

CORE IDEA

BFS explores by distance layers and finds shortest paths in an unweighted graph. DFS follows a path using recursion or an explicit stack and is useful for reachability, components, and structural exploration. Marking visited state prevents repeated work.

HOW IT WORKS



BFS Queue explores by distance / layer.

DFS Stack follows a path before backtracking.

Real-world application

Real-world application: model intersections as vertices and roads as edges. BFS finds the fewest road segments when every edge has equal cost. If travel times differ, use a weighted shortest-path method instead.

CORE + STRETCH PRACTICE

- Core: 9 new problems across trees, grids, and graphs.
Full track: 16 with 7 optional stretch problems.
- Write iterative and recursive traversal; explain width/depth space usage.
- Retest graph setup and visited state before adding ordering patterns.

CURRENT LIBRARY

[PDF - DFS](#)
[PDF - BFS](#)
[PDF - Topological Sort](#)
[VIDEO - BFS](#)

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Can you define visited-state ownership?
- Can you explain why BFS does or does not give the needed guarantee?

DECISION RULE

If graph setup causes more errors than traversal, isolate representation and visited-state drills.

Backtracking, heaps, intervals, and ordering

Make state restoration and ordering decisions visible before implementation.

Weekly outcome

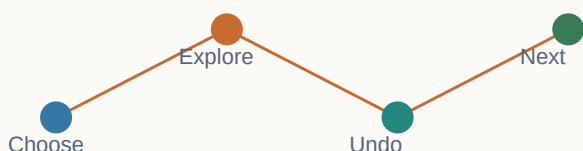
Select the smallest tool that matches the structure: branching choices, priority, overlap, or dependencies.

CORE IDEA

Backtracking explores choices and restores state. A heap exposes the highest- or lowest-priority item. Intervals capture overlap; topological sorting orders a DAG's dependencies.

HOW IT WORKS

BACKTRACKING



ORDERING

earliest

next

later

A heap exposes the next highest-priority item.

Real-world application

Real-world application: build a valid meal bundle by exploring and undoing choices, dispatch the next most urgent order with a heap, and sequence kitchen tasks only after their prerequisites finish.

CORE + STRETCH PRACTICE

- Core: 8 new problems, two per pattern. Full track: 16 with 8 optional stretch problems.
- Name choose/explore/undo; state the heap or interval ordering key.
- Keep one weak pattern active next week; retest it after 48 hours.

CURRENT LIBRARY

- [PDF - Backtracking](#)
- [PDF - Heap Priority Queue](#)
- [PDF - Intervals](#)
- [PDF - Topological Sort](#)

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Can you explain why the chosen ordering is enough?
- Do all backtracking branches restore shared state exactly?

DECISION RULE

Keep one weak advanced pattern active next week; park the others after a clean re-solve.

Dynamic programming and midpoint calibration

Derive states and transitions explicitly, then compare evidence with the baseline.

Weekly outcome

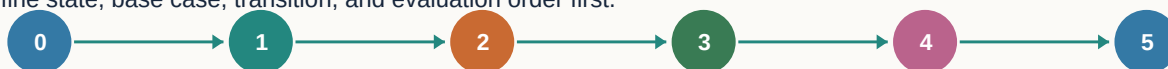
Describe a subproblem precisely before writing a recurrence.

CORE IDEA

Dynamic programming stores answers to overlapping subproblems. Define the state, base case, transition, and evaluation order. A one-dimensional state may summarize a sequence prefix; a two-dimensional state may represent two independent positions or capacities.

HOW IT WORKS

Define state, base case, transition, and evaluation order first.



State i = best valid answer from earlier states + today's choice

Real-world application

Real-world application: choose a sequence of delivery options with the lowest total cost under a constraint. Each state represents the best result up to a position; the transition compares allowed prior choices. State the constraint before writing the formula.

CORE + STRETCH PRACTICE

- Core: 8 new DP problems, four each for 1D/2D. Full track: 14 with 6 optional stretch problems.
- Write state/base/transition/order first; run one 45-minute coding mock (page 18).
- Re-solve five highest-cost errors; carry unfinished core work forward.

CURRENT LIBRARY

- VIDEO** - Dynamic Programming 1d
- PDF** - Dynamic Programming 1d
- PDF** - Dynamic Programming 2d

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Can you derive a recurrence without copying a template?
- Are repaired mock dimensions improving?

DECISION RULE

Escalate a repeated root cause, not one low score. Human Calibration remains a separate, qualification-based path.



System-design framing and core components

Create the first three traceable portfolio write-ups with one consistent decision structure.

Weekly outcome

Start with the user path and a measurable need; introduce components only to meet that need.

CORE IDEA

A design write-up should connect requirements to estimates, API and data ownership, read/write paths, failure behavior, and an explicit trade-off. A load balancer spreads requests across healthy stateless instances; a cache reduces repeated reads but needs freshness rules.

HOW IT WORKS



Draw the request, response, data owner, and asynchronous boundary.

Real-world application

Real-world application: for a food-delivery menu, route requests across healthy APIs, cache low-change menu data, and send checkout writes to the authoritative order store. Inventory and payment ownership must remain explicit.

PRACTICE TARGETS

- Write #1: load-balanced service, including requirements, estimates, API, data, and bottleneck.
- Write #2: cache-backed read path with invalidation, failure behavior, and one rejected alternative.
- Write #3: API-gateway boundary with routing, authorization, rate protection, and observability.
- Defend one alternative per major component and score one write-up.

CURRENT LIBRARY

PDF - Load Balancing

VIDEO - Caching

PDF - API Gateway

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Do you clarify scope before choosing components?
- Can you trace one request end to end?
- Are write-ups #1-#3 independently understandable?

PORTFOLIO CHECKPOINT

Weeks 7-9 build 8 of 10 system-design write-ups. This week completes #1-#3.

DECISION RULE

If structure is weak, repeat the same prompt; if depth is weak, vary the bottleneck.



Data, scale, and failure

Make growth, consistency, and recovery behavior measurable in two new designs.

Weekly outcome

Tie each scaling choice to a measured limit and state the consistency or recovery cost it adds.

CORE IDEA

Partitioning divides data across storage owners but creates routing and hot-key risks. Replication improves read availability but introduces lag and conflict policy. Queues absorb bursts while adding delay, retries, and eventual consistency; dead-letter handling exposes poison messages.

HOW IT WORKS



Measure partition skew and oldest-event age; throttle producers when consumers fall behind.

Real-world application

Real-world application: a restaurant event queue can absorb a dinner rush. Monitor oldest-event age and consumer lag; apply backpressure before the queue grows without bound. Partition by a key that preserves required ordering.

PRACTICE TARGETS

- Write #4: partitioning, consistency, migration, and hot-key handling.
- Write #5: asynchronous processing with bounded retry, dead letters, backpressure, and recovery.
- Estimate traffic, storage, and bandwidth; run one region-down, slow-dependency, or hot-partition drill.

CURRENT LIBRARY

- [PDF - Database Replication](#)
- [PDF - Database Sharding](#)
- [PDF - Message Queues](#)
- [PDF - Consistent Hashing](#)

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Can you explain the consistency consequence of each data choice?
- Can you identify the first scaling limit quantitatively?
- Are write-ups #4-#5 linked to failure drills?

PORTFOLIO CHECKPOINT

Cumulative system-design portfolio: 5 of 10 write-ups (#1-#5).

DECISION RULE

Do not add components unless a stated requirement or measured bottleneck justifies them.



Applied system designs

Add three archetypes and practise changing requirements without losing coherence.

Weekly outcome

Use one clear request path for each archetype, then test how one changed requirement affects it.

CORE IDEA

A URL shortener maps a compact key to a durable destination. Chat systems need conversation ordering and delivery state. Notification systems separate durable intent from provider delivery, retries, and user-visible status.

HOW IT WORKS

- **URL shortener** → key → durable mapping
- **Chat** → ordered conversation events
- **Notifications** → queue → provider → status

Real-world application

Real-world application: receipt delivery can be retried after a timeout. Use a stable delivery ID plus verified provider idempotency; otherwise record unknown outcomes and reconcile. Chat separately needs per-conversation ordering.

PRACTICE TARGETS

- Write #6: URL shortener; #7: chat; #8: notifications, each with a 45-minute first pass.
- Answer two follow-up pivots per design without starting over.
- Run one timed design mock and record the weakest rubric dimension.

CURRENT LIBRARY

- VIDEO** - URL Shortener
- PDF** - Chat System Walkthrough
- VIDEO** - Notification System

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Can you preserve a coherent design when requirements change?
- Do you explain trade-offs before naming technologies?
- Are write-ups #6-#8 saved with follow-up decisions?

PORTFOLIO CHECKPOINT

Cumulative system-design portfolio: 8 of 10 write-ups (#1-#8).

DECISION RULE

Choose the next company-calibrated design from the weakest applied archetype.

Behavioral evidence and US-context translation

Turn career history into distinct, truthful stories a new listener can calibrate.

Weekly outcome

Make your individual decisions and results understandable without inflating title or scope.

CORE IDEA

The STAR structure gives a listener context (Situation), responsibility (Task), personal action (Action), and evidence of outcome (Result). Use it as a guide, not a script; name team contributions accurately and explain unfamiliar role scope plainly.

HOW IT WORKS



Keep your own action distinct from the team's work.

Real-world application

Real-world application: when describing a production incident, state the service impact, your assigned responsibility, the action you personally took, and what changed afterward. Separate measured result from team-wide outcome.

PRACTICE TARGETS

- Build eight distinct STAR stories mapped to 12 competencies.
- Record six answers and remove setup that does not improve understanding.
- Prepare a 90-second explanation of role, company, and scope.

CURRENT LIBRARY

- VIDEO** - STAR Method
- PDF** - Question Bank
- TEMPLATE** - Behavioral Story Bank
- TEMPLATE** - Title Translation Guide

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Does every story separate your actions from the team's?
- Can a listener calibrate scope without inflated title translation?

PORTFOLIO CHECKPOINT

No new system-design write-up this week. Preserve the verified total of 8 of 10.

DECISION RULE

Replace vague stories with specific evidence; do not add adjectives.



Company calibration and complete interview loops

Finish the ten-write-up portfolio inside two evidence-bounded practice loops.

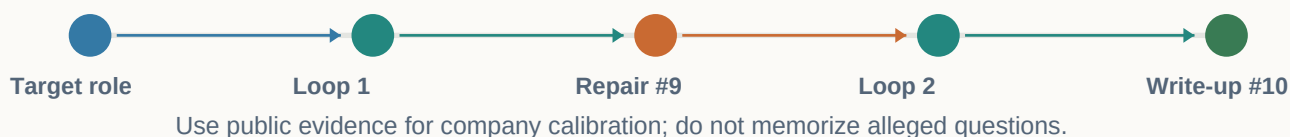
Weekly outcome

Adapt preparation to the role using credible public evidence, not memorized claims about questions.

CORE IDEA

A full loop tests whether skills remain consistent across formats and time pressure. Use the first loop to find a repairable weakness, record the corrected design as write-up #9, then test the repair in a separate loop before writing #10.

HOW IT WORKS



Real-world application

Real-world application: compare two target role descriptions and public engineering material. Select only preparation topics that the evidence supports; write down what remains unknown rather than treating online anecdotes as guarantees.

PRACTICE TARGETS

- Select evidence-approved preparation banks for two target companies.
- Run two full interview loops on separate days.
- Repair the system-design round from each loop as write-ups #9 and #10.
- Repair other misses within 24 hours; repeat only failed segments.

CURRENT LIBRARY

- PDF** - Interview Loop Gap Map
- PDF** - Final Loop Interview Day Runbook
- PDF** - Evidence Comparison Correction Workbook
- TEMPLATE** - Recruiter Screen Notes

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Are two consecutive mock loops near your chosen threshold?
- Can you adapt examples without memorizing alleged questions?
- Do write-ups #9-#10 include decisions, failure path, and repair?

PORTFOLIO CHECKPOINT

System-design portfolio complete: 10 of 10 write-ups (#1-#10).

DECISION RULE

Drop any company bank without adequate public evidence; do not pad it with generic guesses.

Stabilize, apply, and protect interview performance

Consolidate proven behaviors, confirm the portfolio, and reduce avoidable day-of risk.

Weekly outcome

Finish with reliable retrieval and clear logistics, not a last-minute expansion of the syllabus.

CORE IDEA

Spaced retrieval tests whether a skill is available without warm-up. A final full loop checks transitions between interview types. Lowering practice volume before the event preserves recovery time; it does not replace sleep or preparation.

HOW IT WORKS



Reduce volume before the interview; protect sleep, setup, and recovery time.

Real-world application

Real-world application: treat the interview day like a planned trip. Confirm time zone, route or video setup, materials, and buffer time before the event so a preventable logistics failure does not consume attention.

PRACTICE TARGETS

- Re-solve the top 12 errors from a blank editor.
- Run one final full loop, then reduce volume for 48 hours.
- Review all 10 write-ups and rehearse the weakest design's opening, bottleneck, and failure path.
- Confirm time zones, equipment, role scope, and contract-model questions.

CURRENT LIBRARY

- [PDF](#) - Question Bank
- [PDF](#) - Final Loop Interview Day Runbook
- [PDF](#) - Compensation And Offer Decision Workbook
- [TEMPLATE](#) - Interview Availability
- [TEMPLATE](#) - Thank You

Open links after eligible Library sign-in.

CHECKPOINT SELF-ASSESSMENT

- Is performance stable across coding, design, and behavioral rounds?
- Can you recover aloud when stuck?
- Are all 10 write-ups easy to retrieve?

PORTFOLIO CHECKPOINT

Final verified system-design total: 10 of 10. Week 12 adds no required new write-up.

DECISION RULE

Use the final week to stabilize proven skills, not chase novel hard problems.

Evidence to leave with

Completion means the artifacts exist, can be retrieved, and can be explained. The roadmap does not guarantee an interview outcome.

The evidence portfolio

- DSA**
42 core new problems; 75 with optional stretch. Keep spaced re-solves and an error log.
- System design**
10 named write-ups: Weeks 7-9 create #1-#8; Week 11 creates #9-#10.
- Mocks**
At least six timed attempts across interview types, with failed segments repaired.
- Behavioral**
Eight reusable stories with personal action, evidence, and an honest scope boundary.
- Company prep**
Two target-company passes grounded in adequate public evidence.

SYSTEM-DESIGN WRITE-UP COUNT

W07

3 write-ups

W08

2 write-ups

W09

3 write-ups

W11

2 write-ups

10 complete

FINAL CONSISTENCY CHECK

- Every checklist item points to an artifact or observable performance result.
- All ten system-design write-ups are numbered and traceable to the named weeks.
- The next action targets the weakest repeated dimension instead of adding random volume.

READINESS GUARDRAIL

Finishing the plan is not itself a readiness signal. Repaired attempts and stable performance provide the evidence. Adjust the next step to the gap that repeats.

Use pages 16-18 for the workload budget, eight-week route, and observable checkpoint rules.

Protect four focused sessions.

Four 105-minute sessions give 7 hours; reserve up to 2 more core hours for unfinished work or repair.

Session	105-minute core block	Up to 30-minute core buffer
1 - learn + start	30 min concept; 55 min attempt; 20 min evidence log.	Finish an incomplete core attempt.
2 - apply	65 min attempts; 25 min inspect; 15 min recall.	Inspect and repair an unresolved miss.
3 - test	45 min timed mock/design; 30 min review; 30 min repair.	Complete mock review or a focused repair.
4 - retrieve	45 min spaced re-solves; 40 min targeted repair; 20 min checkpoint.	Retrieve unstable core skills; carry unfinished work forward.

CODING TARGETS - COUNT NEW PROBLEMS SEPARATELY FROM RE-SOLVES

Week	Core new	Optional stretch	Full-track new
W02 - maps / prefixes / pointers	9	+6	15
W03 - window / binary search	8	+6	14
W04 - trees / graphs	9	+7	16
W05 - branching / ordering	8	+8	16
W06 - dynamic programming	8	+6	14
TOTAL	42	+33	75

Core is not a mastery shortcut. Log attempted, independently solved, and repaired problems separately. A timed attempt that stalls is evidence to repair, not a solved-problem tick. Use the buffer for core first; carry work forward if needed. Stretch starts only after core and repair, with additional time or weeks beyond this budget.

Keep ten design write-ups: W07 #1-#3, W08 #4-#5, W09 #6-#8, W11 #9-#10. Use a 45-minute first pass and a short repair pass; an implementable prototype is optional stretch. W10 retains eight behavioral stories, and W12 consolidates without adding new designs.

Compress only after a baseline.

Use this route if basic coding is already fluent and the diagnostic supports combining topics.

Slot / source	Four 105-minute core sessions	Evidence to retain
01 / W01-02	S1: two diagnostics + error log S2: maps/prefix attempts + inspect S3: pointers + six raw stories S4: repair + spaced re-test	6 new problems: 2 per pattern. Baseline first; keep conditions comparable.
02 / W03-04	S1: 4 search/window attempts S2: graph setup + 3 attempts S3: 3 graph attempts + coding mock S4: inspect + repair + re-test	10 new: 4 search/window + 6 traversal. Graph representation precedes ordering.
03 / W05	S1: 2 backtracking attempts S2: 2 heap attempts + inspect S3: interval + topological attempt S4: retrieval + repair	6 new: 2 backtracking, 2 heap, 1 interval, 1 topological. Restore state; explain ordering.
04 / W06	S1: DP state + 2 attempts S2: 3 attempts + inspect S3: coding mock + 3 attempts S4: five re-solves + repair	8 new: 4 each 1D/2D. State/base/transition/order written first.
05 / W07-08	S1: learn + design #1 S2: 45-min first passes #2-#3 S3: design #4 + failure review S4: design #5 + repair/re-test	Five designs, not prototypes. Preserve ownership, transaction, and recovery decisions.
06 / W09-10	S1: design #6 + story work S2: design #7 + story work S3: design #8 + story work S4: six recorded answers + repair	Three designs; complete eight truthful stories using the six raw drafts.
07 / W11	S1: two-company evidence + prep S2: full loop on day A S3: full loop on day B S4: review + repairs #9-#10	Each loop: coding 45, design 45, behavioral 15 min. Repair the saved design artifacts.
08 / W12	S1: top-error retrieval + log S2: final 105-minute loop S3: ten-design review + logistics S4: focused repair + buffer	Keep all ten write-ups. Finish before the 48-hour taper; protect sleep.

REDUCED VOLUME, PRESERVED PREREQUISITES

This adaptation reduces coding from 42 core new problems to **30** (6 + 10 + 6 + 8), removes optional videos/prototypes, and keeps **all ten design write-ups**. Reaching the full-track 75 needs 45 additional problems and extra time/weeks. The 12-week core/stretch table still totals $42 + 33 = 75$.

Re-test after at least 48 hours. If a block cannot fit 105-135 minutes, carry it forward and extend the calendar. If basic arrays/maps or traversal are not fluent, use the twelve-week route. A shorter calendar is not a readiness promise.

Attempt. Inspect. Repair. Re-test.

Score applicable dimensions only; N/A is not zero and totals across formats are not comparable.

Coding - 45 minutes: Given an integer array and k, find the longest contiguous range with at most k distinct values. Clarify empty input and k=0; explain the invariant, implement, test edges, and state complexity. Use 5/25/10/5 minutes for clarify/code/test/explain.

Behavioral - 20 minutes: Describe a decision you owned under uncertainty. Answer in 2 minutes; probe alternatives, personal contribution, and evidence for 5 minutes; inspect the recording for 5 minutes; repair and re-answer for 8 minutes.

System design - 45 minutes: open the food-delivery mock and six-dimension rubric in the active [Timed Technical Rehearsal Workbook](#). A timer and self-review are enough; optional AI/coaching access is separate.

Dimension / format	0 - blocked	1 - partial	2 - observable control
Recognition / DSA	No valid framing.	Plausible framing with prompts.	States the pattern and why it fits.
Correctness / DSA	Wrong result.	Happy path; misses an edge.	Tests edges and explains the invariant.
Complexity / DSA	Wrong or absent bound.	Correct big-O; weak justification.	Derives time/space from operations and input size.
Evidence / behavioral	Vague or inflated ownership.	Personal action clear; weak result.	Truthful action, alternative, observable result, and scope boundary.
Communication / DSA + behavioral	Cannot expose reasoning.	Understandable with prompting.	Concise reasoning; states uncertainty and recovers aloud.

Applicable maximum: DSA uses four dimensions, 8 points; behavioral uses two, 4 points. Mark other roadmap dimensions N/A and exclude them: maximum = 2 x applicable dimensions. System design uses the rehearsal workbook six-dimension scorecard, 12 points; never combine totals.

Practice gate: repair any 0. Advance after two comparable attempts at least 48 hours apart with DSA correctness or behavioral evidence at 2 and other applicable dimensions at least 1. For design, ownership, duplicate safety, and failure handling must each score 2; other dimensions at least 1 on both attempts. Safety gates take priority over the author-set 10/12 first-pass goal.

Record: prompt/date, scores or N/A, one evidence quote, repair, success condition, and re-test date. These are practice rules, not validated hiring thresholds. Keep optional product access separate.