PRACTICAL

INTERVIEW RUNTIME

iOS INTERVIEW PREPARATION

Swift, Concurrency, System Design, and
Mock Interviews for Senior and Staff Engineers.

MIKE SALARI

Interview Runtime

iOS Interview Preparation

Swift, Concurrency, System Design, and Mock Interviews for Senior and Staff Engineers

- **Release:** September 10, 2026
- **Edition:** First Interview Runtime edition
- **Author:** Mike Salari, 15+ years in mobile engineering

Who this book is for

This book is for Senior iOS Engineers with a real loop on the calendar, roughly two to eight weeks out, who need a preparation system rather than another list of questions. It is equally built for engineers targeting staff, tech-lead, architect, or engineering-management scope, where the interview measures mechanisms and judgment more than syntax.

It is not for first-job seekers. If you are preparing for your first iOS role, the fundamentals reference in Part 5 and the free diagnostic will serve you better than the full route, and they will tell you honestly when the rest of the book becomes worth your time.

What changes after 30 days

If you run the route as written, you finish with concrete artifacts, not a feeling:

- A diagnosed profile: your scored baseline across six behavioral dimensions.
- A set of scored recordings, one per practice cycle, measured against the same anchors.
- A story bank of leadership and decision evidence, retrievable under a two-minute clock.
- Completed simulations from the simulation lab, run under realistic constraints.
- Current-platform answers checked against the stable baseline declared in the release manifest.

The honest promise: this is a preparation system, never a guaranteed outcome. Hiring depends on the role, company, interviewers, market, and your performance on a particular day. What the book can do is make your improvement visible and your evidence easy for an interviewer to score.

How to read it

First: do not be scared of the page count. This is a system with a reference attached, not a novel. Nobody reads it cover to cover, and it was not designed to be read that way. Use the table of contents to jump straight to your route, your weak areas, and the simulations your target role requires; the field reference in Part 5 exists for the days you need one answer fast, not for reading in sequence.

Start with the diagnostic, then pick a route in Chapter 3: a 7-day compressed pass for an imminent loop, a 14-day focused route, or the full 30-day system. Each route names the chapters it uses and the artifacts it produces, so you never read a page without knowing what it is for.

Which parts are for your level

- **Junior (0-2 years):** the fundamentals reference (chapter 25) plus the diagnostic. Run the 7-day route only when a real screen is booked; most of Parts 1-3 will earn its place later.
- **Mid-level (2-4 years):** the diagnostic, Part 1 in full, chapter 12 (live coding), and chapter 25 as the gap-filler. The senior calibration blocks show you the next level's bar.
- **Senior iOS Engineer:** the core audience. The 14-day route, Parts 1 and 2, and simulations 01-08.
- **Staff / Tech Lead:** the 30-day route, Parts 1-3 with emphasis on chapters 13 and 17, and simulations 06-10.
- **Mobile Architect:** chapter 18 as the spine; chapters 7, 8, 11, and 13 as the evidence; simulations 05, 06, and 08.
- **Engineering Manager / Delivery lead:** chapters 19-20, chapter 14 for the story bank, and simulations 09-12; skim Part 1 for currency rather than depth.

Where this guidance comes from

The guidance in this book is based on the author's 500+ screening-side technical interviews conducted and reviewed across consumer, enterprise, fintech, and regulated environments; on 15+ years of production mobile engineering; on primary-source

platform documentation checked against the declared stable baseline; and on original composite scenarios written for this book. It is not based on confidential question banks, company-specific leaks, or any private hiring material. Where frequency language appears ("commonly," "often"), it is bounded by that experience, not by a formal public dataset.

How to use this edition

Interview Runtime is maintained as a canonical, versioned manuscript. Technical claims and code are reviewed against the stable platform baseline declared in the release manifest. Material labeled **Beta Watch** is dated, may change, and must not be treated as a production requirement.

Companion scorecards, routes, canvases, simulations, and repositories use the same version number as the book. If an asset version differs, retrieve the matching release before using it with a chapter.

Copyright and professional-use notice

This publication is educational material, not legal, employment, security, or financial advice. Company and interview scenarios are original composites unless explicitly identified and permissioned. Do not upload proprietary employer code, confidential interview questions, personal candidate recordings, or private hiring material to public services.

Code examples are teaching material. Validate platform availability, security assumptions, error handling, performance, accessibility, and product requirements before using an idea in production.

Errata, accessibility, and updates

The final release provides a versioned errata channel, accessible digital formats, and a dated update log. Every storefront claim must match the release record for its source, counts, toolchain, and validation results.

Contents

About Mike Salari

15+ years building production mobile systems

Mike Salari is an iOS engineer, mobile technical leader, and author with 15+ years of experience building and shipping production mobile products. He has led mobile teams and interviewed engineers from early-career screens through staff and engineering-management loops.

This book grew out of a simple frustration with most interview material: a list of questions does not prepare someone for the conversation that actually happens in a senior round. The candidate needs to make assumptions visible, reason about production consequences, adapt under constraint, and show evidence of judgment.

Interview Runtime reflects the preparation method Mike uses in practice: diagnose the gap, practise the signal under a timer, score observable behavior, correct one thing, and repeat. The routes, scorecards, simulations, canvases, and drills exist to make that improvement visible rather than motivational.

Mike writes and publishes at salari.dev.

Contact

- Website and resources: https://salari.dev
- Books and editions: https://mikesalari.dev
- Email: mike@salari.dev

Questions about the material, corrections, and accessibility issues are welcome. Technical corrections are especially valued: include the chapter and the relevant source so the next edition can improve quickly.

CHAPTER 1

Start Here: Turn Experience into Interview Evidence

The outcome of this chapter

In approximately fifteen minutes, you will define the role you are pursuing, map the rounds you expect, record one untreated baseline answer, and choose the next action. Do this before reading technical chapters. Otherwise, familiarity can disguise the gap between recognizing a good answer and producing one under observation.

What senior interviews actually need to see

An interview is a short, imperfect sampling system. Your years of experience are context; they are not the evidence being scored. Interviewers can only evaluate what becomes visible in the round.

For an experienced iOS candidate, strong evidence usually appears in six forms:

1. **Correctness:** your Swift, memory, concurrency, and platform reasoning is sound.
2. **Production judgment:** you connect a solution to lifecycle, failure, testing, performance, accessibility, security, privacy, rollout, and maintenance when relevant.
3. **System reasoning:** you turn ambiguous requirements into explicit constraints and defensible boundaries.
4. **Execution:** you clarify, plan, implement, test, recover, and use the clock well.
5. **Scope:** your examples show ownership, influence, and decisions appropriate to the role.
6. **Communication:** the interviewer can follow your assumptions, trade-offs, and conclusion.

A candidate can be excellent at the job and weak at exposing this evidence in forty-five minutes. This book does not ask you to manufacture a persona. It helps you retrieve, structure, and rehearse real evidence so the interviewer does not have to guess.

Create your interview map

Write the best information you currently have. Unknown is a valid answer; it becomes a question for the recruiter.

Field	Your answer
Target role and level	
Company or company type	
Interview date	
Recruiter/hiring-manager screen	Known / likely / unknown
Swift or fundamentals round	Known / likely / unknown
Live coding environment	Xcode / shared editor / unknown
Mobile system design	Known / likely / unknown
Architecture or past-project deep dive	Known / likely / unknown
Behavioral/leadership rounds	Known / likely / unknown
Take-home	Known / likely / unknown
AI assistance policy	Allowed / restricted / prohibited / unknown
Your available preparation hours	

If three or more round details are unknown, your first task is not another chapter. Ask the recruiter for the loop, duration, tools, evaluation areas, and AI policy. You are not asking for confidential questions; you are asking for the format in which you will be evaluated.

Record the untreated baseline

Set a two-minute timer. Do not outline first. Record audio locally or write a timed transcript for this prompt:

Baseline prompt: Tell me about the most consequential technical decision you made on an iOS product. What made it difficult, what did you decide, and what happened afterward?

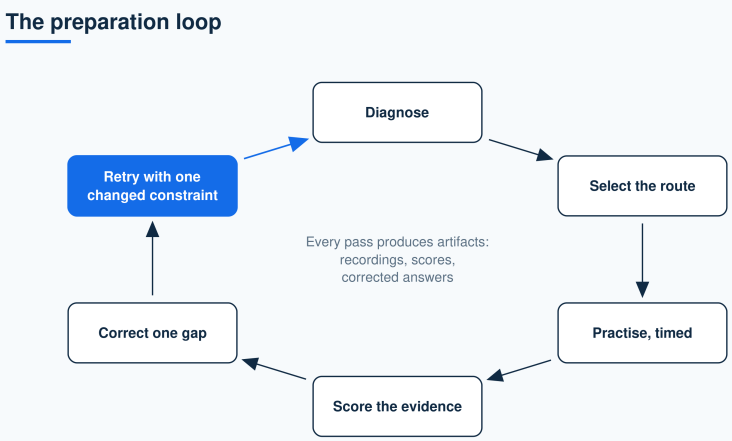
Preserve this attempt. Do not delete it because you dislike it. The discomfort is information. You will repeat the prompt near the end of your route.

Immediately afterward, answer these questions without rewriting the response:

- Did I state the product or engineering stakes?
- Was my personal responsibility unambiguous?
- Did I name at least two plausible options?
- Did I explain constraints rather than declare one pattern “best”?
- Did I describe how the decision was validated?
- Did I state a real result, an honest proxy, or explicitly say what was not measured?
- Could an interviewer ask what *I* did without exposing a vague “we”?
- Did I finish with a clear conclusion inside two minutes?

Count the checks only as observations. This is not your final score; Chapter 2 supplies the full rubric.

The preparation loop



Use the same loop throughout the book:

Work the questions in order: **diagnose, then select, then learn, then practise, then score, then correct, then repeat**.

Diagnose

Attempt a prompt before studying it. Record what you can produce now.

Select

Prioritize by target round, consequence, and weakness. Do not read every page in order merely because it exists.

Learn

Study a mental model and a decision framework. Avoid memorizing a paragraph that collapses when the interviewer changes one constraint.

Practise

Produce something observable: a spoken answer, diagram, code, test, story, or decision memo.

Score

Use behavioral anchors. “Felt better” is too vague to guide the next attempt.

Correct

Choose one binding weakness. For example: missing clarification, wrong isolation model, unbounded scope, no test, or an unsupported result.

Repeat

Attempt the same prompt with a changed constraint. Improvement is the ability to adapt, not the ability to recite the correction.

Practice modes

Solo

Record locally, respect the timebox, score only after finishing, and write one correction. Solo practice is useful for volume but vulnerable to generous self-scoring.

Peer

Give the peer the public prompt and rubric. Ask them to write evidence before assigning a number. Compare scores and discuss differences greater than one point.

Interview simulation

Use a full brief, uninterrupted timing, follow-up questions, and a formal debrief. Do not stop mid-answer to teach. Feedback comes after the attempt.

Rules for working with AI

AI can generate variations, challenge assumptions, summarize a transcript you are permitted to share, or act as a practice partner. It can also confidently provide obsolete platform guidance or code that violates isolation, cancellation, security, or API contracts.

For every AI-assisted artifact:

1. State the interview or privacy rule.
2. Protect proprietary and personal information.
3. Verify the output against primary sources and executable tests where possible.
4. Be able to explain every submitted line and decision.
5. Disclose assistance when the process requires it.

During a real interview, comply with the company's explicit policy. If the policy is unclear, ask. Undisclosed assistance is not a preparation technique; it is an integrity risk.

Your first decision

Continue to Chapter 2 and complete the diagnostic if you have at least forty-five minutes. If your interview is within seven days, complete the shortened diagnostic in Chapter 2 and then enter the 7-Day Fast Track. If you are missing basic Swift or iOS fundamentals, the diagnostic will route you to the field reference without forcing every reader through it.

Artifact completed: target-role map and untreated baseline answer.

Next action: complete the Senior iOS Interview Scorecard.

CHAPTER 2

Diagnose: The Senior iOS Interview Scorecard

The outcome of this chapter

You will complete a forty-five-minute baseline across six dimensions, attach evidence to each rating, identify the three gaps most likely to affect your target loop, and prescribe the next drill. The scorecard is a preparation instrument, not a hiring prediction.

Scoring without false precision

Rate each dimension from 1 to 5. A score describes observable behavior in one attempt under stated conditions. It does not describe your worth, total career ability, or probability of receiving an offer.

- **1: Unsafe or absent:** a serious misconception, no workable approach, or behavior that creates material risk.
- **2: Partial:** relevant knowledge appears, but prompting is necessary and important constraints remain unaddressed.
- **3: Solid:** the core approach is correct, common trade-offs are explained, and a workable decision is reached for the target round.
- **4: Strong:** constraints are anticipated, production implications are visible, and the conclusion is clear and supported.
- **5: Leading:** ambiguity is reframed, second-order effects are compared, organizational scope is handled, and the answer creates a reusable decision mechanism.

Use the role calibration column. A 5 is not “more facts.” It is consistently better judgment and leverage for the scenario.

Dimension 1: Swift, memory, and concurrency correctness

Most senior rejections I have debriefed did not come from a missing fact. They came from a candidate discovering a gap live, in minute twelve of a round, with an

interviewer watching the recovery. The table below exists so that discovery happens here instead, on paper, where a low score costs nothing and buys you a named drill. Read each row as a prediction of what an interviewer would write about you today, then score honestly; the plan you build in the next chapter is only as good as the numbers you feed it.

Score	Observable behavior
1	States an unsafe ownership or isolation model; cannot explain the failure it creates.
2	Defines concepts but misses lifetime, cancellation, reentrancy, `Sendable`, or mutation implications without prompting.
3	Chooses a correct model, explains core ownership/isolation, and handles a normal failure path.
4	Anticipates races, cancellation, actor/lifetime boundaries, testing, and migration cost.
5	Adapts the model across legacy and modern code, identifies second-order performance/API effects, and proposes an incremental enforcement strategy.

Red flags: "actors make code thread-safe" without discussing reentrancy; detached tasks as a default; `weak` everywhere; suppressing concurrency warnings without an isolation argument; confusing value semantics with automatic cheap copies.

Correction drills: ownership diagram; callback-to-async bridge; actor-reentrancy trace; strict-concurrency migration plan.

Dimension 2: iOS platform judgment and production engineering

Score	Observable behavior
1	Gives an API-shaped answer that cannot survive lifecycle, failure, privacy, or delivery constraints.
2	Knows common APIs but needs prompts to discuss testing, observability, accessibility, compatibility, or recovery.
3	Connects the implementation to lifecycle, failure, tests, and an appropriate platform constraint.
4	Prioritizes reliability, performance, privacy/security, accessibility, rollout, and measurement according to risk.

Score	Observable behavior
5	Designs an evolvable platform strategy, separates policy from implementation, and explains organizational ownership and migration.

Red flags: framework-name answers; treating App Review as the deployment plan; storing tokens in ordinary preferences; full-resolution images in feed memory; a permission request without product context.

Correction drills: release-risk checklist; privacy/data-flow map; performance hypothesis; state-ownership diagram.

Dimension 3: Mobile system design and trade-offs

Score	Observable behavior
1	Starts drawing components without understanding the user flow or constraints.
2	Produces a plausible happy-path diagram but misses state ownership, offline behavior, failure, or evolution.
3	Clarifies goals, defines state/data flow, allocates responsibilities, and handles common failure modes.
4	Compares alternatives across consistency, performance, security/ privacy, observability, migration, and rollout.
5	Reframes the highest-risk decision, sequences reversible steps, defines ownership/metrics, and adapts the architecture as scale or organization changes.

Red flags: backend system design with a phone icon; unspecified source of truth; “use a cache” without invalidation; conflict resolution omitted; no migration or release path.

Correction drills: one-page requirements canvas; state/sync canvas; failure walk; staged evolution plan.

Dimension 4: Live-coding process and test discipline

Score	Observable behavior
1	Codes immediately, remains opaque, cannot reach a runnable core, or declares completion without checking behavior.
2	Reaches part of the solution but loses time to scope, silence, premature abstraction, or reactive debugging.

Score	Observable behavior
3	Clarifies, states a plan, implements a correct core, traces/tests normal and edge behavior, and communicates progress.
4	Controls time, makes deliberate simplifications, recovers from mistakes, and tests the highest-risk behavior.
5	Negotiates scope, exposes invariants, selects high-value tests, and evolves the solution without destabilizing the core.

Red flags: eight-minute silence; architecture before requirements; no concrete example; test-last declaration with no time reserved; accepting generated code that cannot be explained.

Correction drills: five-minute plan-only exercise; verbal trace; core-first implementation; bug-recovery rehearsal.

Dimension 5: Leadership, scope, and communication

Score	Observable behavior
1	Cannot identify personal responsibility or describes authority, activity, and opinion instead of influence and result.
2	Gives a real story, but stakes, options, mechanism, result, or learning remain vague.
3	States scope and responsibility, compares options, explains action/influence, and reports an honest result.
4	Shows cross-functional judgment, handles disagreement, measures consequences, and changes a team mechanism afterward.
5	Demonstrates durable leverage across teams or systems, with explicit trade-offs, adoption strategy, and reflection on limits.

Red flags: permanent “we”; invented precision; blaming another function; conflict resolved only by escalation; strategy described as a list of technologies.

Correction drills: responsibility rewrite; six-story bank; follow-up stress test; technical strategy one-pager.

Dimension 6: Interview execution

Score	Observable behavior
1	Does not answer the question, ignores constraints, or cannot recover interaction with the interviewer.
2	Relevant content appears, but the answer is unstructured, overlong, assumption-heavy, or dependent on repeated prompts.
3	Clarifies when needed, signposts the answer, manages time, states assumptions, and concludes directly.
4	Adapts depth to interviewer signals, surfaces uncertainty honestly, and handles follow-ups without losing structure.
5	Improves the problem framing collaboratively, makes complex reasoning easy to inspect, and creates room for productive follow-up.

Red flags: answering a different question; uninterrupted monologue; false certainty; asking ten low-value clarifications; never making a decision.

Correction drills: 30-second/2-minute/8-minute answer ladder; assumption-first response; summary-first replay.

The forty-five-minute diagnostic

Use a timer. Do not pause to research. Record locally or preserve written work.

Exercise A: Swift and concurrency: 8 minutes

Prompt: A legacy UIKit screen starts two callback-based requests, combines their results, and updates the UI. It occasionally shows stale state after navigation and now produces strict-concurrency warnings. Explain how you would diagnose and migrate it without rewriting the entire feature at once.

Score Dimensions 1, 2, and 6.

Exercise B: Mobile system design: 12 minutes

Prompt: Design the iOS client for a field-work app that must edit records offline for an entire day, synchronize when connectivity returns, and preserve an audit trail when two users edit the same record.

Produce a simple diagram and decision summary. Score Dimensions 2, 3, and 6.

Exercise C: Live coding: 12 minutes

Prompt: Given an asynchronous search service, design or implement the core logic that waits briefly after typing, prevents stale results from appearing, exposes loading/error state, and can be tested without real network calls.

If twelve minutes is insufficient for full code, state scope and finish a testable core. Score Dimensions 1, 4, and 6.

Exercise D: Leadership: 8 minutes

Prompt: Tell me about a time you opposed a technical direction supported by influential peers or leadership. How did you decide whether to push, test, compromise, or commit?

Score Dimensions 5 and 6.

Evidence and scoring: 5 minutes

For each dimension, record:

Dimension	Score	Exact evidence observed	Highest-risk gap	Next drill
Swift/memory/ concurrency				
Platform/production				
System design				
Live coding				
Leadership/scope				
Interview execution				

Evidence must be specific: “I named cancellation before prompting,” “I had no conflict policy,” or “I spent seven minutes coding before testing.” “Good” and “bad” are judgments, not evidence.

Choose the three priority gaps

Use this order:

1. **Safety/correctness:** any score of 1 in Dimensions 1-4 comes first.
2. **Target-round relevance:** prioritize dimensions used in the confirmed loop.
3. **Execution bottleneck:** if knowledge is present but invisible, practise the container.

4. **Leverage:** select a gap whose correction improves multiple rounds.

Do not prioritize by embarrassment. A weak obscure topic matters less than a repeated inability to clarify, test, or explain trade-offs.

Worked scoring example

Suppose a candidate's offline design names Core Data, a REST API, and a retry queue. They draw screens and services quickly. After prompting, they add timestamps but still do not define the source of truth, idempotency, conflict policy, audit representation, or behavior when a record is deleted on one device and edited on another.

A defensible score is:

- Platform judgment: **2**: useful APIs, important production constraints require prompts.
- System design: **2**: plausible happy path, missing consistency and conflict decisions.
- Execution: **3** if the answer was structured and responsive; **2** if the candidate drew immediately and depended on the interviewer to discover every requirement.

The correction is not “study Core Data.” It is: complete the state/sync canvas, choose a conflict policy, define idempotent operations, walk through edit/edit and delete/edit cases, then repeat the design with a new constraint.

Interpreting the baseline

- A score of 1 is a blocking misconception or execution failure for that scenario.
- A score of 2 identifies a prompt-dependent gap.
- A score of 3 is a workable baseline, not permission to stop practising.
- A score of 4 is strong evidence when repeated across changed constraints.
- A score of 5 should be rare and supported by observable leverage, not senior vocabulary.

Do not convert the average into an “offer readiness percentage.” Preserve the dimension profile. Your weakest consequential dimension often controls the outcome more than the average.

Final reassessment rule

At the end of your selected route, repeat equivalent prompts without reading your original answer. Score them using the same anchors. Record what changed, what

INTERVIEW RUNTIME 1.0 · IOS

Your first attempt is only the baseline.

The full edition turns interview knowledge into evidence you can inspect, correct, retry, and defend under pressure.

- The complete iOS interview preparation book
- PDF and EPUB access
- 7, 14, and 30-day preparation routes
- Diagnostics, practice cycles, and simulations
- Access to the Interview Runtime dashboard

Buy the full iOS edition

interviewruntime.com

Interview preparation, not a guarantee of employment or interview outcomes.