Pulse — SwiftUI End-to-End Learning Roadmap

This document is an offline, step-by-step mentor guide for building the Pulse app. Each step includes: concepts, reviewer questions, and concrete TODOs describing the exact code you are expected to write in the Pulse project.

Step 1: Project & App Structure

Key Topics

- SwiftUI App lifecycle
- App protocol and WindowGroup
- Folder and module organization
- Preview system

Reviewer / Reasoning Questions

- Why does SwiftUI not need AppDelegate by default?
- What responsibilities belong in PulseApp?

Pulse Project — TODOs

- TODO: Create PulseApp.swift using @main and App
- TODO: Set FeedView as the root view inside WindowGroup
- TODO: Create folder structure for Views, Models, ViewModels, Services
- TODO: Add at least one SwiftUI Preview

Step 2: Core View Building

Key Topics

- View protocol
- Stacks and layout primitives
- Modifiers and order
- Reusable views

Reviewer / Reasoning Questions

- What is the single responsibility of this view?
- Why does modifier order matter here?

Pulse Project — TODOs

- TODO: Implement ActivityRowView with VStack/HStack
- TODO: Use Text and Spacer to layout content
- TODO: Apply padding and font modifiers intentionally

- TODO: Ensure ActivityRowView has no state

Step 3: State Management Fundamentals

Key Topics

- @State usage
- View invalidation
- Why views are structs

Reviewer / Reasoning Questions

- Who owns this state?
- What triggers a re-render?

Pulse Project — TODOs

- TODO: Add @State activities array to FeedView
- TODO: Simulate loading activities locally
- TODO: Trigger state changes via Button or onAppear
- TODO: Verify UI refreshes correctly

Step 4: Data Flow Between Views

Key Topics

- @Binding basics
- Data down, actions up
- Closures for events

Reviewer / Reasoning Questions

- Why is a closure better than a binding here?
- Who owns the data being mutated?

Pulse Project — TODOs

- TODO: Pass Activity into ActivityRowView as immutable data
- TODO: Add a tap or button in ActivityRowView
- TODO: Pass a closure from FeedView to handle row actions
- TODO: Avoid using @Binding for navigation or selection

Step 5: Lists & Dynamic Views

Key Topics

- List and ForEach
- Identifiable models

Reviewer / Reasoning Questions

- Why must list items be identifiable?
- What happens when identity changes?

Pulse Project — TODOs

- TODO: Conform Activity to Identifiable
- TODO: Render activities using List and ForEach
- TODO: Support swipe-to-delete
- TODO: Update state and verify row animations

Step 6: Navigation

Key Topics

- NavigationStack
- NavigationLink
- Passing data through navigation

Reviewer / Reasoning Questions

- Where should navigation state live?
- What should not be passed through navigation?

Pulse Project — TODOs

- TODO: Wrap FeedView in NavigationStack
- TODO: Navigate from row to ActivityDetailView
- TODO: Pass selected Activity to detail screen
- TODO: Avoid embedding navigation logic inside rows

Step 7: User Input

Key Topics

- TextField and forms
- Focus management
- Validation

Reviewer / Reasoning Questions

- Who owns form state?
- When should validation run?

Pulse Project — TODOs

- TODO: Create EditActivityView with a Form
- TODO: Bind TextField values to local @State
- TODO: Add basic validation before saving
- TODO: Implement Cancel / Save actions

Step 8: View Composition & Architecture

Key Topics

- MVVM responsibilities
- Avoiding fat views

Reviewer / Reasoning Questions

- Why should views not contain business logic?
- What makes a ViewModel testable?

Pulse Project — TODOs

- TODO: Create FeedViewModel
- TODO: Move business logic out of FeedView
- TODO: Expose state via published properties
- TODO: Inject ViewModel into FeedView

Step 9: Observable State & Data Models

Key Topics

- ObservableObject
- @Published
- StateObject lifecycle

Reviewer / Reasoning Questions

- Who owns the ObservableObject?
- What happens if it is recreated?

Pulse Project — TODOs

- TODO: Mark FeedViewModel as ObservableObject
- TODO: Use @StateObject in FeedView
- TODO: Use @ObservedObject in child views if needed
- TODO: Avoid unnecessary @Published properties

Step 10: Async & Concurrency

Key Topics

- async/await
- Tasks and cancellation

Reviewer / Reasoning Questions

- When is a task cancelled?
- Why prefer .task over onAppear?

Pulse Project — TODOs

- TODO: Fetch activities using async function
- TODO: Call async load from .task modifier
- TODO: Handle cancellation correctly
- TODO: Update UI on the main actor

Step 11: Networking & API Fetching

Key Topics

- URLSession
- Codable
- Dependency injection

Reviewer / Reasoning Questions

- Why abstract networking behind protocols?
- How do you test failures?

Pulse Project — TODOs

- TODO: Define ActivityService protocol
- TODO: Implement JSONPlaceholderActivityService
- TODO: Decode GET /posts response
- TODO: Inject service into ViewModel

Step 12: View Lifecycle Awareness

Key Topics

- task(id:)
- Pull-to-refresh

Reviewer / Reasoning Questions

- Why can onAppear be unreliable?

- How do you prevent duplicate loads?

Pulse Project — TODOs

- TODO: Add .refreshable to FeedView
- TODO: Reload data on pull-to-refresh
- TODO: Prevent duplicate concurrent loads

Step 13: Animations

Key Topics

- Implicit animations
- Transitions

Reviewer / Reasoning Questions

- What exactly is being animated?
- Why might animation not run?

Pulse Project — TODOs

- TODO: Animate list insertions and deletions
- TODO: Use withAnimation for state changes
- TODO: Add a simple transition for empty state

Step 14: Conditional UI & State-driven Design

Key Topics

- Loading, empty, error states

Reviewer / Reasoning Questions

- Which state drives this branch?
- How do you avoid flicker?

Pulse Project — TODOs

- TODO: Introduce loading and error states in ViewModel
- TODO: Render different UI based on state
- TODO: Add a reusable LoadingView

Step 15: Environment & App-wide State

Key Topics

- EnvironmentObject

- Global app state

Reviewer / Reasoning Questions

- When is global state justified?
- What risks does it introduce?

Pulse Project — TODOs

- TODO: Create AppSettings model
- TODO: Inject AppSettings via environment
- TODO: Read environment values in views

Step 16: Persistence

Key Topics

- UserDefaults
- Simple caching

Reviewer / Reasoning Questions

- What should never be persisted here?
- When should persistence occur?

Pulse Project — TODOs

- TODO: Persist a simple user preference
- TODO: Restore preference on app launch
- TODO: Keep persistence logic out of views

Step 17: Debugging & Best Practices

Key Topics

- Debugging SwiftUI
- Performance

Reviewer / Reasoning Questions

- How do you debug over-rendering?
- What are common SwiftUI pitfalls?

Pulse Project — TODOs

- TODO: Add debug logging to ViewModel
- TODO: Use previews to test states
- TODO: Identify and remove unnecessary state

Step 18: Polishing & Production Readiness

Key Topics

- Reusable styles
- Architecture cleanup

Reviewer / Reasoning Questions

- What would you refactor before release?
- What technical debt remains?

Pulse Project — TODOs

- TODO: Extract reusable modifiers
- TODO: Standardize spacing and fonts
- TODO: Clean up unused code paths

Step 19: Unit Testing

Key Topics

- ViewModel testing
- Async tests

Reviewer / Reasoning Questions

- What should not be unit tested?
- How do you test async failures?

Pulse Project — TODOs

- TODO: Write unit tests for FeedViewModel
- TODO: Mock ActivityService
- TODO: Test success and failure cases

Step 20: UI Testing

Key Topics

- XCUITest
- Navigation testing

Reviewer / Reasoning Questions

- What makes UI tests flaky?
- Which flows are worth testing?

Pulse Project — TODOs

- TODO: Add accessibility identifiers
- TODO: Write UI test for Feed → Detail navigation
- TODO: Test loading and error flows

Step 21: Professional Testing Strategy

Key Topics

- Test pyramid
- CI tradeoffs

Reviewer / Reasoning Questions

- What do senior engineers avoid testing?
- Where is the highest ROI?

Pulse Project — TODOs

- TODO: Review test coverage vs confidence
- TODO: Decide which tests to remove or keep
- TODO: Optimize test execution time