Complete iOS Development Learning Outline: Swift & SwiftUI 2026

Overview

This comprehensive year-long curriculum transforms you from a complete beginner to a publishing-ready iOS developer using Swift 6 and SwiftUI. Designed for AI-assisted learning with 1-3 minute video lessons, each task is small enough to explain and code quickly while building toward real portfolio projects. With 365 days and approximately 260 learning days (5 days/week), you'll progress through 260-520 micro-lessons organized into phases, modules, and daily tasks.

Phase 1: Swift Fundamentals (Weeks 1-8, ~40 days)

Module 1.1: Getting Started (Days 1-3)

Day 1: Environment Setup

- Task 1: Install Xcode 16 from Mac App Store
- Task 2: Create your first Playground project
- Task 3: Write "Hello, World!" and understand the console output

Day 2: Variables and Constants

- Task 1: Declare variables with `var` keyword
- Task 2: Declare constants with `let` keyword
- Task 3: Understand when to use `var` vs `let`

Day 3: Data Types Basics

- Task 1: Work with String type and string interpolation

- Task 2: Work with Int and Double numeric types
- Task 3: Work with Bool type and understand true/false

Module 1.2: Collections and Control Flow (Days 4-10)

Day 4: Arrays

- Task 1: Create and initialize arrays[1]
- Task 2: Access array elements with subscript syntax
- Task 3: Add, remove, and modify array elements

Day 5: Dictionaries

- Task 1: Create dictionaries with key-value pairs
- Task 2: Access and modify dictionary values
- Task 3: Use optional binding for safe dictionary access

Day 6: Sets

- Task 1: Create sets and understand uniqueness
- Task 2: Perform set operations (union, intersection)
- Task 3: Compare sets vs arrays use cases

Day 7: If Statements

- Task 1: Write basic if-else conditionals
- Task 2: Use comparison operators (==, !=, <, >)
- Task 3: Combine conditions with && and | |

Day 8: Switch Statements

- Task 1: Write switch statements with multiple cases
- Task 2: Use ranges in switch cases
- Task 3: Understand switch exhaustiveness

Day 9: Loops - For

- Task 1: Loop through arrays with for-in
- Task 2: Loop through ranges (0...10)
- Task 3: Use enumerated() for index and value

Day 10: Loops - While

- Task 1: Write while loops with conditions
- Task 2: Write repeat-while loops
- Task 3: Use break and continue keywords

Module 1.3: Functions (Days 11-15)

Day 11: Function Basics

- Task 1: Define functions with func keyword
- Task 2: Call functions and understand execution flow
- Task 3: Return values from functions

Day 12: Function Parameters

- Task 1: Add parameters to functions
- Task 2: Use multiple parameters
- Task 3: Provide default parameter values

Day 13: Argument Labels

- Task 1: Use external argument labels
- Task 2: Omit argument labels with underscore
- Task 3: Understand when to use clear labels

Day 14: Return Values

- Task 1: Return single values
- Task 2: Return tuples for multiple values
- Task 3: Use named tuple elements

Day 15: Function Practice

- Task 1: Build a temperature converter function
- Task 2: Build a BMI calculator function
- Task 3: Build a tip calculator function

Module 1.4: Closures (Days 16-18)

Day 16: Closure Syntax

- Task 1: Write basic closure expressions[2]
- Task 2: Understand closure syntax vs functions
- Task 3: Use closures as function parameters

Day 17: Trailing Closures

- Task 1: Use trailing closure syntax
- Task 2: Apply map() to arrays with closures
- Task 3: Apply filter() to arrays with closures

Day 18: Capture Values

- Task 1: Understand closure capturing
- Task 2: Use escaping closures
- Task 3: Practice with common closure patterns

Module 1.5: Structs and Classes (Days 19-28)

Day 19: Struct Basics

- Task 1: Define your first struct[2]
- Task 2: Add properties to structs
- Task 3: Create struct instances

Day 20: Struct Methods

- Task 1: Add methods to structs
- Task 2: Use self keyword inside methods
- Task 3: Understand mutating methods

Day 21: Computed Properties

- Task 1: Create computed properties with get
- Task 2: Add set to computed properties
- Task 3: Build a Person struct with full name computed property

Day 22: Property Observers

- Task 1: Use willSet observer
- Task 2: Use didSet observer
- Task 3: Apply observers to track property changes

Day 23: Initializers

- Task 1: Write custom init methods
- Task 2: Use memberwise initializers
- Task 3: Provide default values in initializers

Day 24: Class Basics

- Task 1: Define classes vs structs[2]
- Task 2: Understand reference vs value types
- Task 3: Create class instances

Day 25: Inheritance

- Task 1: Create subclasses with inheritance[2]
- Task 2: Override methods in subclasses
- Task 3: Use super keyword

Day 26: Class Initializers

- Task 1: Write designated initializers
- Task 2: Write convenience initializers
- Task 3: Understand initializer inheritance

Day 27: Access Control

- Task 1: Use private, fileprivate, internal
- Task 2: Use public and open
- Task 3: Apply access control to properties

Day 28: Struct vs Class Decision

- Task 1: Build example with struct
- Task 2: Build example with class
- Task 3: Understand when to choose each

Module 1.6: Optionals (Days 29-33)

Day 29: Optional Basics

- Task 1: Understand what optionals are
- Task 2: Declare optional variables with ?
- Task 3: Understand nil values

Day 30: Optional Binding

- Task 1: Use if let for safe unwrapping
- Task 2: Use guard let for early exits
- Task 3: Compare if let vs guard let

Day 31: Optional Chaining

- Task 1: Chain optional properties
- Task 2: Chain optional methods
- Task 3: Use optional chaining with arrays

Day 32: Nil Coalescing

- Task 1: Use ?? operator for default values
- Task 2: Chain multiple ?? operators
- Task 3: Apply to real scenarios

Day 33: Force Unwrapping

- Task 1: Understand ! operator
- Task 2: Know when force unwrapping is safe
- Task 3: Avoid common force unwrap crashes

Module 1.7: Protocols and Enums (Days 34-40)

Day 34: Enum Basics

- Task 1: Define basic enums[2]
- Task 2: Use enum cases
- Task 3: Switch on enum values

Day 35: Associated Values

- Task 1: Add associated values to enum cases[2]
- Task 2: Extract associated values with switch
- Task 3: Build a Result enum example

Day 36: Raw Values

- Task 1: Define enums with raw values
- Task 2: Initialize enums from raw values
- Task 3: Compare associated vs raw values

Day 37: Protocol Basics

- Task 1: Define protocols with required properties[2]
- Task 2: Define protocols with required methods
- Task 3: Conform types to protocols

Day 38: Protocol Extensions

- Task 1: Extend protocols with default implementations
- Task 2: Add methods to protocol extensions
- Task 3: Understand protocol-oriented programming

Day 39: Codable Protocol

- Task 1: Conform structs to Codable[1]
- Task 2: Encode to JSON data
- Task 3: Decode from JSON data

Day 40: Swift Fundamentals Project

- Task 1: Build a Todo model with structs and protocols
- Task 2: Add encoding/decoding functionality
- Task 3: Test with sample data in Playground

Phase 2: SwiftUI Basics (Weeks 9-16, ~40 days)

Module 2.1: First SwiftUI App (Days 41-45)

Day 41: SwiftUI Project Setup

- Task 1: Create first SwiftUI app project[3][1]
- Task 2: Explore project structure (App, ContentView)
- Task 3: Understand SwiftUI preview canvas

Day 42: Text Views

- Task 1: Display text with Text view[4][1]
- Task 2: Style text with font, color, weight
- Task 3: Use multiline text alignment

Day 43: Image Views

- Task 1: Display SF Symbols with Image[5][4]
- Task 2: Add custom images to Assets
- Task 3: Apply image modifiers (resizable, aspectRatio)

Day 44: Stacks - VStack

- Task 1: Arrange views vertically with VStack[4][5]
- Task 2: Control spacing and alignment
- Task 3: Nest VStacks for complex layouts

Day 45: Stacks - HStack and ZStack

- Task 1: Arrange views horizontally with HStack[4]
- Task 2: Layer views with ZStack
- Task 3: Combine stacks for complex layouts[1]

Module 2.2: Views and Modifiers (Days 46-52)

Day 46: Colors and Gradients

- Task 1: Apply Color to backgrounds
- Task 2: Create LinearGradient
- Task 3: Use RadialGradient and AngularGradient

Day 47: Frames and Padding

- Task 1: Set view frames with width/height
- Task 2: Apply padding modifiers
- Task 3: Understand frame vs padding

Day 48: Shapes

- Task 1: Use Rectangle, Circle, Capsule shapes
- Task 2: Apply fill and stroke modifiers
- Task 3: Create custom shape combinations

Day 49: Buttons

- Task 1: Create Button with action closure[6][4]
- Task 2: Style button labels
- Task 3: Use ButtonStyle protocols

Day 50: Modifiers Order

- Task 1: Understand modifier order matters
- Task 2: Apply background before padding vs after
- Task 3: Practice modifier composition

Day 51: Custom Modifiers

- Task 1: Create ViewModifier protocol implementations
- Task 2: Use custom modifiers with .modifier()
- Task 3: Build reusable style modifiers

Day 52: Mini Project - Profile Card

- Task 1: Design profile card UI with Image and Text
- Task 2: Style with gradients and shadows
- Task 3: Make card reusable as custom view

Module 2.3: Lists and Forms (Days 53-60)

Day 53: List Basics

- Task 1: Create static List with fixed items[5]
- Task 2: Add sections to lists
- Task 3: Style list rows

Day 54: ForEach

- Task 1: Loop through data with ForEach[5]
- Task 2: Use identifiable data
- Task 3: Understand id parameter

Day 55: Dynamic Lists

- Task 1: Create array of data models
- Task 2: Display dynamic list from array[5]
- Task 3: Update list when data changes

Day 56: List Styling

- Task 1: Use listStyle modifiers
- Task 2: Customize row appearance
- Task 3: Add dividers and backgrounds

Day 57: Form Basics

- Task 1: Create Form container
- Task 2: Add sections to forms
- Task 3: Group related inputs

Day 58: Form Inputs - TextField

- Task 1: Add TextField for text input
- Task 2: Bind to @State variable
- Task 3: Style and customize text fields

Day 59: Form Inputs - Toggle and Picker

- Task 1: Add Toggle switches
- Task 2: Add Picker for selections
- Task 3: Bind form inputs to state

Day 60: Mini Project - Settings Screen

- Task 1: Build settings form with sections
- Task 2: Add various input types
- Task 3: Display saved settings

Module 2.4: Basic State Management (Days 61-68)

Day 61: @State Basics

- Task 1: Declare @State property[6][4][5]
- Task 2: Modify @State in button action
- Task 3: Observe UI updates automatically

Day 62: @State with Different Types

- Task 1: Use @State with Bool for toggles
- Task 2: Use @State with String for text
- Task 3: Use @State with numbers

Day 63: Two-Way Binding

- Task 1: Understand $ syntax for bindings[5]
- Task 2: Bind TextField to @State variable
- Task 3: Bind Slider and Picker

Day 64: @Binding Basics

- Task 1: Pass @Binding to child views[5]
- Task 2: Create two-way communication
- Task 3: Understand @State parent, @Binding child pattern

Day 65: View Composition

- Task 1: Extract subviews as separate structs
- Task 2: Pass data to subviews
- Task 3: Pass bindings to subviews

Day 66: State Practice

- Task 1: Build counter app with increment/decrement
- Task 2: Build simple calculator UI
- Task 3: Build temperature converter

Day 67: onAppear and onDisappear

- Task 1: Use onAppear lifecycle method
- Task 2: Use onDisappear for cleanup
- Task 3: Load data on view appear

Day 68: Mini Project - Interactive Quiz

- Task 1: Build quiz UI with questions
- Task 2: Track score with @State
- Task 3: Show results screen

Module 2.5: Layout System (Days 69-76)

Day 69: Spacer

- Task 1: Use Spacer to push views apart
- Task 2: Control Spacer behavior
- Task 3: Create flexible layouts with Spacers

Day 70: Alignment

- Task 1: Align VStack and HStack contents
- Task 2: Use alignment guides
- Task 3: Create custom alignments

Day 71: GeometryReader

- Task 1: Read available size with GeometryReader
- Task 2: Make responsive layouts
- Task 3: Understand GeometryProxy

Day 72: SafeArea

- Task 1: Understand safe area behavior
- Task 2: Ignore safe area when needed
- Task 3: Design edge-to-edge views

Day 73: ScrollView

- Task 1: Create vertical ScrollView
- Task 2: Create horizontal ScrollView
- Task 3: Detect scroll position

Day 74: LazyVStack and LazyHStack

- Task 1: Use LazyVStack for performance
- Task 2: Use LazyHStack for horizontal scrolling
- Task 3: Compare List vs ScrollView with LazyVStack

Day 75: Grid Layout

- Task 1: Create LazyVGrid with columns
- Task 2: Create LazyHGrid with rows
- Task 3: Build photo grid layout

Day 76: Mini Project - Photo Gallery UI

- Task 1: Create grid of placeholder images
- Task 2: Make images tappable
- Task 3: Add smooth scroll behavior

Module 2.6: Basic Animations (Days 77-80)

Day 77: Implicit Animations

- Task 1: Add .animation() modifier
- Task 2: Animate @State changes
- Task 3: Experiment with animation curves[7]

Day 78: Explicit Animations

- Task 1: Use withAnimation block[7]
- Task 2: Control animation timing
- Task 3: Use spring animations[7]

Day 79: Transitions

- Task 1: Add .transition() modifier
- Task 2: Use built-in transitions
- Task 3: Combine transitions

Day 80: Mini Project - Animated Button

- Task 1: Create pulsing animation
- Task 2: Add tap animation feedback
- Task 3: Combine multiple animations

Phase 3: Navigation & Advanced State (Weeks 17-24, ~40 days)

Module 3.1: Navigation Basics (Days 81-88)

Day 81: NavigationStack Basics

- Task 1: Wrap views in NavigationStack[1][5]
- Task 2: Add navigation title
- Task 3: Customize navigation bar

Day 82: NavigationLink

- Task 1: Create NavigationLink to new views[1][5]
- Task 2: Pass data to destination views
- Task 3: Style navigation links

Day 83: Programmatic Navigation

- Task 1: Use NavigationPath for programmatic navigation
- Task 2: Push views with path
- Task 3: Pop back with path

Day 84: Navigation Bar Buttons

- Task 1: Add toolbar items
- Task 2: Add leading and trailing buttons
- Task 3: Handle button actions

Day 85: Sheet Presentation

- Task 1: Present sheet modals[1]

- Task 2: Dismiss sheets programmatically
- Task 3: Pass data to sheets

Day 86: FullScreenCover

- Task 1: Present full screen modals
- Task 2: Understand sheet vs fullScreenCover
- Task 3: Handle dismiss actions

Day 87: Alert Dialogs

- Task 1: Show alert with title and message
- Task 2: Add alert buttons and actions
- Task 3: Use @State to control alert

Day 88: Mini Project - Master-Detail App

- Task 1: Create list of items
- Task 2: Navigate to detail on tap
- Task 3: Show item details with toolbar

Module 3.2: TabView and More Navigation (Days 89-94)

Day 89: TabView Basics

- Task 1: Create TabView with multiple tabs
- Task 2: Add tab icons and labels
- Task 3: Customize tab appearance

Day 90: Tab Selection

- Task 1: Control tab selection with @State
- Task 2: Switch tabs programmatically
- Task 3: Handle tab change events

Day 91: Complex Navigation

- Task 1: Nest NavigationStack inside TabView
- Task 2: Handle navigation within tabs
- Task 3: Reset navigation on tab switch

Day 92: Confirmation Dialog

- Task 1: Show confirmation dialog
- Task 2: Add multiple action buttons
- Task 3: Handle user choices

Day 93: Custom Navigation

- Task 1: Hide default navigation bar
- Task 2: Build custom navigation UI
- Task 3: Handle back navigation

Day 94: Mini Project - Tabbed App Shell

- Task 1: Create app with 3 tabs (Home, Search, Profile)
- Task 2: Add navigation in each tab
- Task 3: Style tab bar

Module 3.3: @Observable and Environment (Days 95-105)

Day 95: @Observable Macro Intro

- Task 1: Create @Observable class[8][9][7]
- Task 2: Understand iOS 17+ observation
- Task 3: Compare to old ObservableObject

Day 96: @Observable Properties

- Task 1: Add observable properties[8]
- Task 2: Update properties and see UI refresh
- Task 3: Understand automatic tracking[8]

Day 97: Using @Observable in Views

- Task 1: Declare @Observable instance in view
- Task 2: Access properties directly
- Task 3: Observe changes automatically[8]

Day 98: Sharing @Observable

- Task 1: Pass @Observable instance to child views
- Task 2: Modify data from any view
- Task 3: See changes everywhere

Day 99: @Environment Basics

- Task 1: Use @Environment property wrapper
- Task 2: Access environment values
- Task 3: Understand common environment keys

Day 100: Custom Environment Values

- Task 1: Create custom environment key
- Task 2: Inject custom environment values
- Task 3: Access in child views

Day 101: Environment Object Pattern

- Task 1: Create shared app state with @Observable
- Task 2: Inject at app root with .environment()
- Task 3: Access in any child view

Day 102: @State vs @Observable

- Task 1: Use @State for simple view-local state[9]
- Task 2: Use @Observable for shared app state[9]
- Task 3: Understand when to use each

Day 103: Multiple @Observable Classes

- Task 1: Create separate data models
- Task 2: Inject multiple environment objects
- Task 3: Keep concerns separated

Day 104: @Observable Best Practices

- Task 1: Keep @Observable classes focused
- Task 2: Avoid massive "god objects"
- Task 3: Structure app state logically

Day 105: Mini Project - Todo App with @Observable

- Task 1: Create TodoStore as @Observable
- Task 2: Add, edit, delete todos from any view
- Task 3: Persist state throughout app

Module 3.4: Advanced UI Components (Days 106-112)

Day 106: Custom Views

- Task 1: Extract complex views as custom components
- Task 2: Pass configuration parameters
- Task 3: Make views reusable

Day 107: ViewBuilder

- Task 1: Understand @ViewBuilder attribute
- Task 2: Create custom container views
- Task 3: Accept view content as parameters

Day 108: PreferenceKey

- Task 1: Understand child-to-parent communication
- Task 2: Create custom PreferenceKey
- Task 3: Read preferences in parent

Day 109: Custom Shapes

- Task 1: Conform to Shape protocol
- Task 2: Implement path(in:) method
- Task 3: Create triangle, star, custom shapes

Day 110: Canvas Drawing

- Task 1: Use Canvas for custom drawing
- Task 2: Draw paths, circles, text
- Task 3: Animate canvas content

Day 111: Matched Geometry Effect

- Task 1: Use matchedGeometryEffect modifier
- Task 2: Animate views between positions
- Task 3: Create hero animations

Day 112: Mini Project - Custom Card Component

- Task 1: Build reusable card view
- Task 2: Accept custom content with ViewBuilder
- Task 3: Add animation effects

Phase 4: Networking & APIs (Weeks 25-30, ~30 days)

Module 4.1: URLSession Basics (Days 113-120)

Day 113: URLSession Intro

- Task 1: Understand URLSession architecture[1]
- Task 2: Create URLSession.shared instance
- Task 3: Understand data, download, upload tasks

Day 114: First API Request

- Task 1: Create URL from string
- Task 2: Make GET request with URLSession
- Task 3: Handle completion and data

Day 115: Async/Await Networking

- Task 1: Use async/await for network calls[10][9]
- Task 2: Handle errors with try/catch
- Task 3: Call async functions from Task{}[9]

Day 116: Decoding JSON - Basics

- Task 1: Create Codable model matching JSON[4]
- Task 2: Use JSONDecoder to decode
- Task 3: Handle decoding errors

Day 117: Decoding Nested JSON

- Task 1: Decode JSON with nested objects
- Task 2: Decode JSON arrays
- Task 3: Handle optional fields

Day 118: Custom Decoding

- Task 1: Use CodingKeys enum for custom keys
- Task 2: Implement custom init(from decoder:)
- Task 3: Transform data during decoding

Day 119: Network Error Handling

- Task 1: Handle network errors with Result type
- Task 2: Check HTTP status codes
- Task 3: Show user-friendly error messages

Day 120: Mini Project - Weather API Integration

- Task 1: Get free weather API key
- Task 2: Create Weather model and decode response
- Task 3: Display weather data in UI

Module 4.2: API Service Layer (Days 121-128)

Day 121: Network Manager Pattern

- Task 1: Create NetworkManager class
- Task 2: Extract common request logic
- Task 3: Make reusable fetch methods

Day 122: Generic Network Methods

- Task 1: Create generic fetch method with Codable
- Task 2: Handle different response types
- Task 3: Reuse for multiple endpoints

Day 123: Request Building

- Task 1: Build URLRequest with headers
- Task 2: Add query parameters
- Task 3: Set HTTP method (GET, POST, etc.)

Day 124: POST Requests

- Task 1: Create POST request body
- Task 2: Encode Codable to JSON
- Task 3: Send data to API

Day 125: Authentication Headers

- Task 1: Add Authorization headers
- Task 2: Add API key to requests
- Task 3: Handle authenticated endpoints

Day 126: Loading States

- Task 1: Track loading with @State bool
- Task 2: Show ProgressView during load
- Task 3: Handle success and error states

Day 127: Image Loading

- Task 1: Fetch image data from URL
- Task 2: Convert data to UIImage
- Task 3: Display in AsyncImage view

Day 128: Mini Project - News API App

- Task 1: Integrate news API (NewsAPI.org)
- Task 2: Display articles in list
- Task 3: Show article details

Module 4.3: Advanced Networking (Days 129-142)

Day 129: AsyncImage

- Task 1: Use AsyncImage for remote images
- Task 2: Handle loading placeholders
- Task 3: Handle failed image loads

Day 130: Caching Images

- Task 1: Understand URLCache
- Task 2: Configure cache policies
- Task 3: Build custom image cache

Day 131: Pagination

- Task 1: Load initial page of data
- Task 2: Detect scroll to bottom
- Task 3: Load next page and append

Day 132: Pull to Refresh

- Task 1: Add refreshable modifier
- Task 2: Reload data on pull
- Task 3: Show refresh indicator

Day 133: Debouncing

- Task 1: Understand debounce concept
- Task 2: Delay search API calls
- Task 3: Cancel previous requests

Day 134: Offline Handling

- Task 1: Detect network reachability
- Task 2: Show offline message
- Task 3: Queue requests for retry

Day 135: API Response Validation

- Task 1: Validate response structure
- Task 2: Handle unexpected formats
- Task 3: Provide fallback data

Day 136-142: Full API Project - Recipe App

- Day 136: Plan recipe app with API (Spoonacular)
- Day 137: Create models for recipes
- Day 138: Build network service layer
- Day 139: Implement recipe list screen
- Day 140: Implement recipe detail screen
- Day 141: Add search functionality
- Day 142: Polish UI and error handling

Phase 5: Data Persistence (Weeks 31-36, ~30 days)

Module 5.1: UserDefaults (Days 143-147)

Day 143: UserDefaults Basics

- Task 1: Store simple values (String, Int, Bool)[1]
- Task 2: Retrieve values with keys
- Task 3: Understand when to use UserDefaults

Day 144: @AppStorage

- Task 1: Use @AppStorage property wrapper
- Task 2: Bind to UserDefaults automatically

- Task 3: Observe changes across views

Day 145: Storing Complex Data

- Task 1: Encode Codable to Data
- Task 2: Store Data in UserDefaults
- Task 3: Decode Data back to model

Day 146: Settings Screen

- Task 1: Build settings UI with @AppStorage
- Task 2: Save user preferences
- Task 3: Apply settings across app

Day 147: Mini Project - Theme Switcher

- Task 1: Save theme preference (light/dark)
- Task 2: Apply theme on app launch
- Task 3: Toggle theme with animation

Module 5.2: SwiftData (Days 148-165)

Day 148: SwiftData Introduction

- Task 1: Understand SwiftData vs Core Data[11][12]
- Task 2: Recognize SwiftData benefits for SwiftUI[11]
- Task 3: Plan to use SwiftData for new projects[11]

Day 149: @Model Macro

- Task 1: Create @Model class[11]
- Task 2: Add properties to models
- Task 3: Understand automatic persistence

Day 150: ModelContainer Setup

- Task 1: Configure ModelContainer in app
- Task 2: Inject container to environment
- Task 3: Understand schema configuration

Day 151: @Query in Views

- Task 1: Fetch data with @Query[11]

- Task 2: Display fetched data in list[11]
- Task 3: Observe automatic updates

Day 152: Inserting Data

- Task 1: Access ModelContext from environment
- Task 2: Insert new model instances
- Task 3: Save changes to SwiftData

Day 153: Updating Data

- Task 1: Modify model properties
- Task 2: Observe automatic persistence
- Task 3: Update UI reactively

Day 154: Deleting Data

- Task 1: Delete model instances from context
- Task 2: Use swipe-to-delete in lists
- Task 3: Handle cascade deletes

Day 155: Filtering Queries

- Task 1: Add predicate to @Query
- Task 2: Filter by property values
- Task 3: Combine multiple conditions

Day 156: Sorting Queries

- Task 1: Add sort descriptor to @Query
- Task 2: Sort by multiple properties
- Task 3: Toggle sort order

Day 157: Relationships - One-to-Many

- Task 1: Define relationship properties
- Task 2: Link related models
- Task 3: Query across relationships

Day 158: Relationships - Many-to-Many

- Task 1: Create many-to-many relationships
- Task 2: Add and remove related items

- Task 3: Display related data

Day 159: Model Versioning

- Task 1: Understand schema migrations[11]
- Task 2: Plan for lightweight migrations[11]
- Task 3: Test migration scenarios

Day 160: SwiftData Best Practices

- Task 1: Keep models focused
- Task 2: Use @Query efficiently
- Task 3: Avoid premature optimization

Day 161-165: Full SwiftData Project - Expense Tracker

- Day 161: Plan expense tracker data models
- Day 162: Create Expense and Category models
- Day 163: Build expense list with @Query
- Day 164: Build add expense form
- Day 165: Add charts to visualize spending

Module 5.3: File Management (Days 166-172)

Day 166: FileManager Basics

- Task 1: Access FileManager.default
- Task 2: Get document directory path
- Task 3: List files in directory

Day 167: Saving Files

- Task 1: Write Data to file
- Task 2: Write String to file
- Task 3: Handle file save errors

Day 168: Loading Files

- Task 1: Read Data from file
- Task 2: Decode data to model
- Task 3: Handle missing files

Day 169: File Operations

- Task 1: Copy files
- Task 2: Move files
- Task 3: Delete files

Day 170: Image Persistence

- Task 1: Save images to documents
- Task 2: Load images from disk
- Task 3: Generate thumbnails

Day 171: Document Picker

- Task 1: Present document picker
- Task 2: Import files from picker
- Task 3: Access picked file data

Day 172: Mini Project - Notes App with Files

- Task 1: Save notes as separate files
- Task 2: List all note files
- Task 3: Edit and delete notes

Phase 6: Testing, Debugging & Polish (Weeks 37-42, ~30 days)

Module 6.1: Debugging Techniques (Days 173-180)

Day 173: Print Debugging

- Task 1: Use print() effectively
- Task 2: Log state changes
- Task 3: Format debug output

Day 174: Xcode Debugger Basics

- Task 1: Set breakpoints
- Task 2: Step through code
- Task 3: Inspect variables

Day 175: LLDB Commands

- Task 1: Use po command

- Task 2: Use expr command
- Task 3: Modify values while debugging

Day 176: View Debugging

- Task 1: Use SwiftUI preview debugging
- Task 2: Inspect view hierarchy
- Task 3: Debug layout issues

Day 177: Memory Debugging

- Task 1: Use memory graph debugger[13][14]
- Task 2: Detect retain cycles
- Task 3: Fix memory leaks

Day 178: Xcode Instruments

- Task 1: Record with Time Profiler
- Task 2: Analyze performance bottlenecks[15]
- Task 3: Optimize slow code

Day 179: Network Debugging

- Task 1: Use network link conditioner
- Task 2: Test slow network scenarios
- Task 3: Debug failed requests

Day 180: Crash Log Analysis

- Task 1: Read crash logs
- Task 2: Symbolicate stack traces[16]
- Task 3: Fix common crashes

Module 6.2: Testing (Days 181-192)

Day 181: Unit Testing Basics

- Task 1: Create first XCTest target[14][1]
- Task 2: Write test case methods
- Task 3: Run tests in Xcode

Day 182: Testing Functions

- Task 1: Test pure functions
- Task 2: Use XCTAssert methods
- Task 3: Test edge cases

Day 183: Testing Models

- Task 1: Test Codable encoding
- Task 2: Test Codable decoding
- Task 3: Test model validation logic

Day 184: Testing ViewModels

- Task 1: Test @Observable state changes
- Task 2: Test business logic
- Task 3: Mock dependencies

Day 185: Mocking

- Task 1: Create mock network service
- Task 2: Inject mocks in tests
- Task 3: Verify mock interactions

Day 186: Async Testing

- Task 1: Test async/await functions
- Task 2: Use expectations for async code
- Task 3: Handle async errors in tests

Day 187: UI Testing Basics

- Task 1: Create UI test target[1]
- Task 2: Record UI test
- Task 3: Run UI tests

Day 188: UI Test Assertions

- Task 1: Query UI elements
- Task 2: Assert element existence
- Task 3: Assert element properties

Day 189: Accessibility Testing

- Task 1: Use Accessibility Inspector[14][16]

- Task 2: Test VoiceOver support
- Task 3: Fix accessibility issues

Day 190: Test Coverage

- Task 1: Enable code coverage
- Task 2: Review coverage report
- Task 3: Add tests for uncovered code

Day 191: Test Best Practices

- Task 1: Keep tests fast and isolated
- Task 2: Use descriptive test names
- Task 3: Organize tests logically

Day 192: CI/CD Introduction

- Task 1: Understand continuous integration
- Task 2: Learn about Xcode Cloud
- Task 3: Plan automated testing

Module 6.3: Performance & Polish (Days 193-202)

Day 193: Performance Best Practices

- Task 1: Avoid expensive view updates
- Task 2: Use LazyVStack appropriately
- Task 3: Profile with Instruments[15]

Day 194: Image Optimization

- Task 1: Resize images appropriately
- Task 2: Use compressed formats
- Task 3: Lazy load images

Day 195: Network Optimization

- Task 1: Batch network requests
- Task 2: Cache responses
- Task 3: Use background tasks

Day 196: Battery Optimization

- Task 1: Minimize location updates
- Task 2: Reduce background processing
- Task 3: Profile energy usage

Day 197: Accessibility

- Task 1: Add accessibility labels
- Task 2: Support Dynamic Type
- Task 3: Test with VoiceOver

Day 198: Localization Basics

- Task 1: Mark strings for localization[16][14]
- Task 2: Export localizable strings
- Task 3: Add Spanish translation example

Day 199: Dark Mode

- Task 1: Support light and dark color schemes
- Task 2: Use semantic colors
- Task 3: Test both modes

Day 200: App Icons and Launch Screen

- Task 1: Design app icon (or generate with AI)
- Task 2: Add icon assets to project
- Task 3: Create launch screen

Day 201: Haptics

- Task 1: Add haptic feedback to buttons
- Task 2: Use appropriate feedback types
- Task 3: Test on device

Day 202: Final Polish

- Task 1: Review all app flows
- Task 2: Fix minor UI issues
- Task 3: Test on multiple devices

Phase 7: Publishing & Advanced Topics (Weeks 43-52, ~40 days)

Module 7.1: App Store Preparation (Days 203-215)

Day 203: App Store Overview

- Task 1: Understand App Store ecosystem
- Task 2: Review App Store guidelines
- Task 3: Plan app store presence

Day 204: Apple Developer Account

- Task 1: Enroll in Apple Developer Program ($99/year)
- Task 2: Complete developer profile
- Task 3: Accept agreements

Day 205: Bundle Identifier

- Task 1: Choose unique bundle identifier
- Task 2: Configure in Xcode project
- Task 3: Register app in App Store Connect[17][18]

Day 206: Certificates and Provisioning

- Task 1: Create distribution certificate[17]
- Task 2: Create App Store provisioning profile
- Task 3: Download and install profiles

Day 207: App Store Connect Record

- Task 1: Create new app record[18][19][17]
- Task 2: Fill in basic information
- Task 3: Configure pricing and availability[19]

Day 208: App Screenshots

- Task 1: Capture screenshots for required sizes[18][17]
- Task 2: Design attractive screenshot layouts
- Task 3: Upload to App Store Connect

Day 209: App Description

- Task 1: Write compelling app description[17]
- Task 2: Add keywords for search[17]
- Task 3: Write promotional text

Day 210: Privacy Policy

- Task 1: Create privacy policy document
- Task 2: Host privacy policy online
- Task 3: Add privacy URL to App Store Connect

Day 211: App Review Information

- Task 1: Provide demo account if needed[17]
- Task 2: Add review notes
- Task 3: Prepare for App Review

Day 212: Archiving the App

- Task 1: Configure release build settings[18][17]
- Task 2: Archive app in Xcode[18][17]
- Task 3: View archive in Organizer

Day 213: Uploading to App Store

- Task 1: Validate archive[17]
- Task 2: Upload to App Store Connect[18][17]
- Task 3: Wait for processing

Day 214: Submitting for Review

- Task 1: Select build in App Store Connect[17]
- Task 2: Complete all required info[17]
- Task 3: Submit for review[19][17]

Day 215: TestFlight Beta

- Task 1: Add internal testers[20][18]
- Task 2: Add external testers
- Task 3: Collect beta feedback[20]

Module 7.2: Advanced SwiftUI (Days 216-230)

Day 216: Advanced Animations

- Task 1: Use keyframe animations
- Task 2: Create physics-based animations[7]
- Task 3: Chain complex animation sequences

Day 217: Custom Transitions

- Task 1: Create custom transition effects
- Task 2: Use AnyTransition
- Task 3: Combine transitions creatively

Day 218: Gesture Recognizers

- Task 1: Use TapGesture, LongPressGesture
- Task 2: Use DragGesture with state[21][22]
- Task 3: Combine multiple gestures

Day 219: Advanced Gesture Handling

- Task 1: Create swipe-to-delete gesture
- Task 2: Create pinch-to-zoom gesture
- Task 3: Create rotation gesture

Day 220: Maps - MapKit

- Task 1: Display Map view[3]
- Task 2: Add annotations to map
- Task 3: Handle map interactions

Day 221: Location Services

- Task 1: Request location permissions[13]
- Task 2: Get user's current location[3]
- Task 3: Update UI with location

Day 222: Camera and Photos

- Task 1: Present camera with UIImagePickerController
- Task 2: Save photos to library
- Task 3: Handle photo permissions

Day 223: Notifications

- Task 1: Request notification permissions
- Task 2: Schedule local notifications[22][21]
- Task 3: Handle notification responses

Day 224: App Extensions

- Task 1: Understand extension types
- Task 2: Create widget extension (basics)
- Task 3: Share data with app groups

Day 225: Combine Framework

- Task 1: Understand Combine basics
- Task 2: Use Publishers and Subscribers
- Task 3: Integrate Combine with SwiftUI

Day 226: Swift Concurrency Deep Dive

- Task 1: Understand actors in detail[23][10]
- Task 2: Use MainActor properly[23][10]
- Task 3: Handle Sendable types[23]

Day 227: Swift 6 Concurrency Safety

- Task 1: Enable strict concurrency checking[24][25][23]
- Task 2: Fix data race warnings[24][23]
- Task 3: Understand Swift 6 safety features[23][9]

Day 228: Advanced SwiftData

- Task 1: Use model inheritance[26]
- Task 2: Optimize complex queries[26]
- Task 3: Track changes with persistent history[26]

Day 229: Custom Metal Shaders

- Task 1: Understand Metal basics[7]
- Task 2: Apply shader to SwiftUI view[7]
- Task 3: Animate shader effects

Day 230: Swift Macros

- Task 1: Use built-in Swift macros[27][9]
- Task 2: Understand macro expansion
- Task 3: Explore macro use cases

Module 7.3: Final Portfolio Projects (Days 231-260)

Days 231-245: Project 1 - Social Media App

- Day 231: Plan features (feed, posts, profiles, likes)
- Day 232: Set up project with navigation and tabs
- Day 233: Create data models with SwiftData
- Day 234: Build authentication UI (mockup)
- Day 235: Build home feed UI
- Day 236: Implement post creation
- Day 237: Build profile screen
- Day 238: Add image upload functionality
- Day 239: Implement likes and comments
- Day 240: Add networking layer (mock API or Firebase)
- Day 241: Implement real-time updates
- Day 242: Add animations and polish
- Day 243: Test all features thoroughly
- Day 244: Fix bugs and optimize performance
- Day 245: Record demo video for portfolio

Days 246-260: Project 2 - E-commerce App

- Day 246: Plan features (products, cart, checkout, orders)
- Day 247: Set up project structure
- Day 248: Create product models
- Day 249: Build product list with API integration
- Day 250: Build product detail screen
- Day 251: Implement shopping cart with SwiftData
- Day 252: Build cart screen with total calculations
- Day 253: Create checkout flow UI
- Day 254: Implement payment mockup (Stripe simulation)
- Day 255: Build order history screen
- Day 256: Add search and filtering
- Day 257: Implement favorites/wishlist
- Day 258: Add animations and transitions
- Day 259: Complete testing and bug fixes
- Day 260: Polish, deploy to TestFlight, update portfolio

Daily Learning Flow (1-3 Minute Video Format)

Each day follows this structure optimized for short videos:

Video Format Template

1. **Introduction (15 seconds)**: "Today we're learning [topic]"
2. **Explanation (30-60 seconds)**: Concept explanation with visual aids
3. **Code Demo (60-90 seconds)**: Live coding in Xcode with AI guidance
4. **Result (15-30 seconds)**: Show the working result in simulator/preview

AI Tutor Integration

- Prompt AI before each video: "Explain [topic] for a 1-minute video"
- Ask AI to break longer tasks: "Split this into three 1-minute tasks"
- Use AI for debugging: "Why is [error] happening?"
- Get next steps: "What should I learn next after [topic]?"

Project-Based Learning Strategy

- Every 3-5 days: Mini-project consolidating recent lessons
- Every 2 weeks: Slightly larger project (1-2 days)
- Every 2 months: Major portfolio project (2-3 weeks)
- All projects use modern 2026 patterns: Swift 6, @Observable, SwiftData, async/await

Success Metrics

Month 3 Checkpoint

- Built 5+ mini apps
- Comfortable with Swift fundamentals
- Can build basic SwiftUI interfaces
- Understand state management basics

Month 6 Checkpoint

- Built 10+ mini apps
- Integrated APIs successfully
- Understand navigation patterns
- Started first complex project

Month 9 Checkpoint

- Completed 2 complex projects
- Using SwiftData for persistence
- Writing tests
- Understanding debugging tools

Month 12 Goal

- Published app on App Store (or ready to publish)
- 3 portfolio projects demonstrating skills
- Comfortable with entire iOS development workflow
- Ready for junior iOS developer positions

Resources and Tools

Essential Tools

- Xcode 16+ (includes Swift 6)[28][23]
- Mac mini M4 (already owned)
- iPhone for testing (already owned)
- Free API keys (weather, news, etc.)
- Apple Developer Account ($99/year for publishing)

Recommended APIs for Projects

- OpenWeatherMap (weather data)
- NewsAPI.org (news articles)
- Spoonacular (recipes)
- TMDB (movies)
- JSONPlaceholder (mock REST API)

Learning Platforms (Reference Only)

- Apple Developer Documentation[29][30]
- Kodeco (Ray Wenderlich)[3]
- Hacking with Swift
- Swift.org Tour
- WWDC Videos (especially 2025)[10][15]

AI Tutor Prompts Library

- "Break [task] into 1-minute steps"
- "Explain [concept] for beginners with Python background"
- "Debug this Swift error: [paste error]"
- "Suggest next micro-task after [completed task]"
- "Review this code for Swift 6 best practices"

Conclusion

This outline provides a complete year-long roadmap from absolute beginner to publishing-ready iOS developer using modern Swift 6 and SwiftUI techniques. Each of the 260 learning days contains 1-3 micro-tasks designed for short video lessons, allowing you to document your learning journey while building real skills. The curriculum emphasizes project-based learning with AI as your tutor, ensuring you gain practical experience building apps that leverage 2026-style iOS development patterns including @Observable state management, Swift 6 concurrency, SwiftData persistence, and modern publishing workflows.[31][12][28][24][10][19][9][8][23][18][7][17][11][1]

By the end of 12 months, you'll have a portfolio of diverse projects demonstrating your ability to build complete, polished iOS applications ready for the App Store.

References

1. SwiftUI Roadmap for iOS Developers - LinkedIn - SwiftUI Learning Roadmap. Below is a step-by-step learning path from zero to super advanced: 1. Unde...
2. Foundations of Swift and SwiftUI - Coursera - Updated in May 2025. ... project. Ideal for aspiring/current app developers to

enhance Swift and Swi...

3. iOS & Swift Learning Paths - Kodeco - Get started with SwiftUI fundamentals, learn how SwiftUI thinks about layout in your app, build grea...
4. SwiftUI Tutorial for Beginners (3.5 hour FULL COURSE) - YouTube - this master class, you'll have a solid understanding how to build SwiftUI ... There's no better plac...
5. SwiftUI for Beginners – Fundamentals to Build Real iOS Apps - ... learning SwiftUI Developers transitioning from UIKit Anyone who wants to build ... How to Learn ...
6. SwiftUI Fundamentals | FULL COURSE | Beginner Friendly - YouTube - My Latest SwiftUI Courses - https://seanallen.teachable.com/?video=b1oC7sLIgpI Swift News Newsletter...
7. SwiftUI 2025: What's Fixed, What's Not, and How I Build Apps Now - A new Observable macro to help improve observation performance. An easy way to construct a Spring an...
8. @Observable in SwiftUI explained – Donny Wals - In this post, we'll explore the new @Observable macro, we'll explore how this macro can be used, and...
9. 7 The Most Useful New Swift Features (2024–2025) - LinkedIn - The focus is clear: less boilerplate, more performance, smarter tools, and cleaner SwiftUI code. 1...
10. What's new in Swift - WWDC25 - Videos - Apple Developer - We'll show examples of Swift adoption throughout more layers of the software stack. Finally, we'll e...
11. SwiftData vs. Core Data: Which Should You Choose in 2025? - While SwiftData is the exciting new frontier, Core Data remains a formidable and mature choice that ...
12. How do you choose between Core Data and SwiftData? I ... - LinkedIn - I personally find that while SwiftData is convenient, and generally speaking works well enough, I fi...
13. IOS App Development Course - India - The Knowledge Academy - Module 1: Introduction to iOS · Module 2: Environment Setup of iOS · Module 3: Objective C · Module ...
14. Swift IOS Development Roadmap Rahul | PDF - Scribd - The document outlines a comprehensive roadmap for Swift iOS development, tailored for an individual ...
15. What's new in SwiftUI - WWDC25 - Videos - Apple Developer - WWDC25 · Better together: SwiftUI and RealityKit · Bring Swift

Charts to the third dimension · Build...

16. Best and No.1 IOS Course Syllabus 2025 - Ficusoft - Our iOS course syllabus covers everything from broadening your technical abilities to providing you ...
17. How to submit an iOS app to the App Store: Step-by-step guide - The process of submitting application to the Apple App Store consists of 5 main steps: Step 1: Creat...
18. Complete iOS App Distribution Guide 2025: App Store, TestFlight ... - The Process in a Nutshell. Publishing on the App Store is a structured, multi-step process. It begin...
19. How to submit an app to the Apple App Store: A Step-By-Step Guide - 1. Initiate The Release Process. To commence with new app publishing, log into iTunes Connect and na...
20. Submitting - App Store - Apple Developer - Submit your apps and games today · Update for new OS releases · Set up in App Store Connect. App Rev...
21. Ios development project ideas - JustAcademy - Beginners might consider creating a simple to-do list app, a weather forecast application, or a basi...
22. Any suggestions for "Practice Projects" to start with iOS development? - Additional ideas could be a flashcards app (so you can use drag gestures), weather app, habit tracke...
23. Why Swift 6 Matters: Key Features for Modern iOS Developers - Swift 6 introduces complete compile-time data race safety, strict concurrency, noncopyable types, ty...
24. Swift 6 Exposes Hidden Concurrency Issues | Juan D. posted on the ... - ... 2025 from Apple Coding Academy, focused on Swift 6.2 and new Strict Concurrency. Swift isn't int...
25. How to update SwiftData fetch when upgrading to Swift 6? - I have a project using Swift 5, and I recently enabled “Strict Concurrency Checking” to “Complete” s...
26. SwiftData and Core Data at WWDC25 - Michael Tsai - Discover how to use class inheritance to model your data. Learn how to optimize queries and seamless...
27. Mastering Swift 6 and SwiftUI in 2025: A Hands-On Tutorial - With new macro support, enhanced concurrency, and declarative UI tools, you're well-equipped to tack...
28. 2025 Roadmap: The Essential iOS Development Skills You Must Have - The iOS landscape is changing. Stay ahead of the curve

with our 2025 roadmap, detailing the critical...

29. Learn to code - Apple Developer - Develop in Swift Tutorials. Learn how to build great-looking apps with Swift and SwiftUI, explore ma...
30. SwiftUI Tutorials | Apple Developer Documentation - Learn how to use SwiftUI to compose rich views out of simple ones, set up data flow, and build the n...
31. If I Had to Start iOS Development in 2026, Here's Exactly What I'd Do - New APIs, new features, and new platform capabilities arrive in SwiftUI first — sometimes only in Sw...