Mobile Programming

iOS System Structure

Lecturer: Ali Najimi
Faculty of Computer Engineering
Sharif University of Technology
Fall 2027

iOS Architecture | Lifecycle | Sandbox | IPA

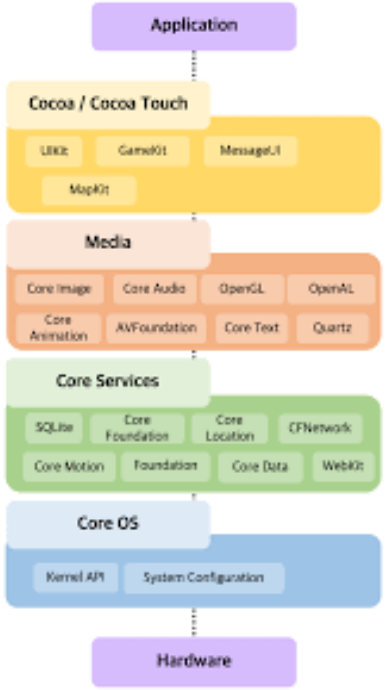


Table of Contents

1. The iOS Architecture Layers
2. Darwin & the XNU Kernel
3. Core OS & System Frameworks
4. Core Services Layer
5. Cocoa Touch Framework
6. The Application Layer & App Sandbox
7. Info.plist: The Control Center
8. The IPA Bundle Structure
9. Deep Dive: View Controller Lifecycle
10. Summary: System Thinking

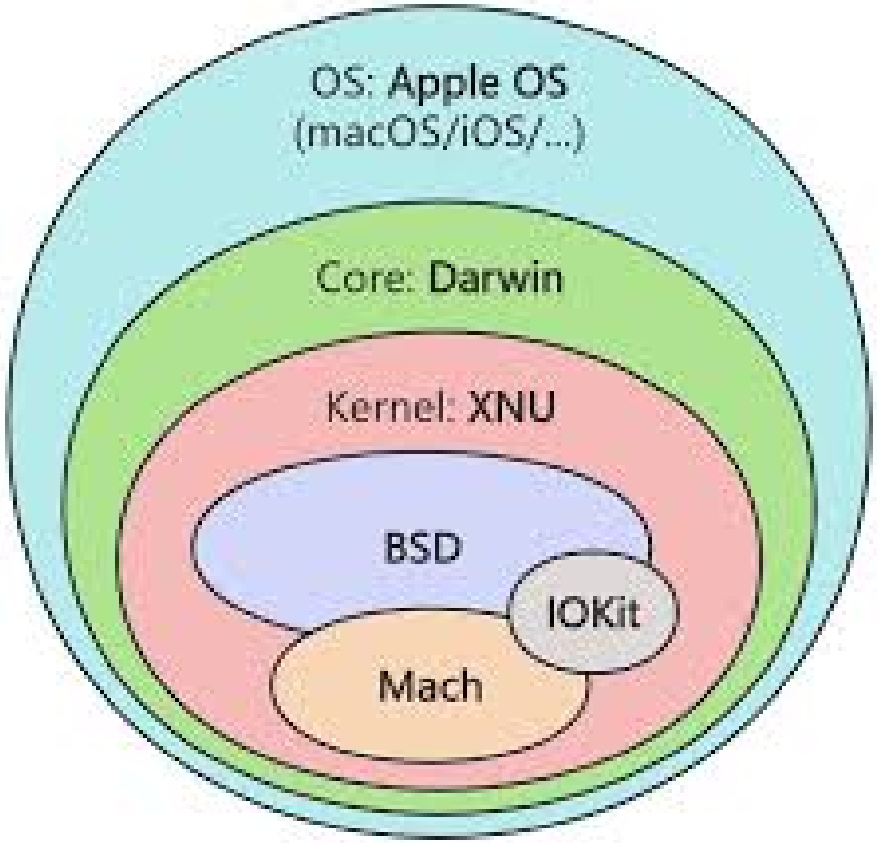
The iOS Architecture Layers

iOS is a **layered architecture** of software components. Understanding this stack is crucial.



Key Takeaway: Your app sits at the top, but it relies heavily on the layers below — from SwiftUI down to the kernel.

Darwin & the XNU Kernel



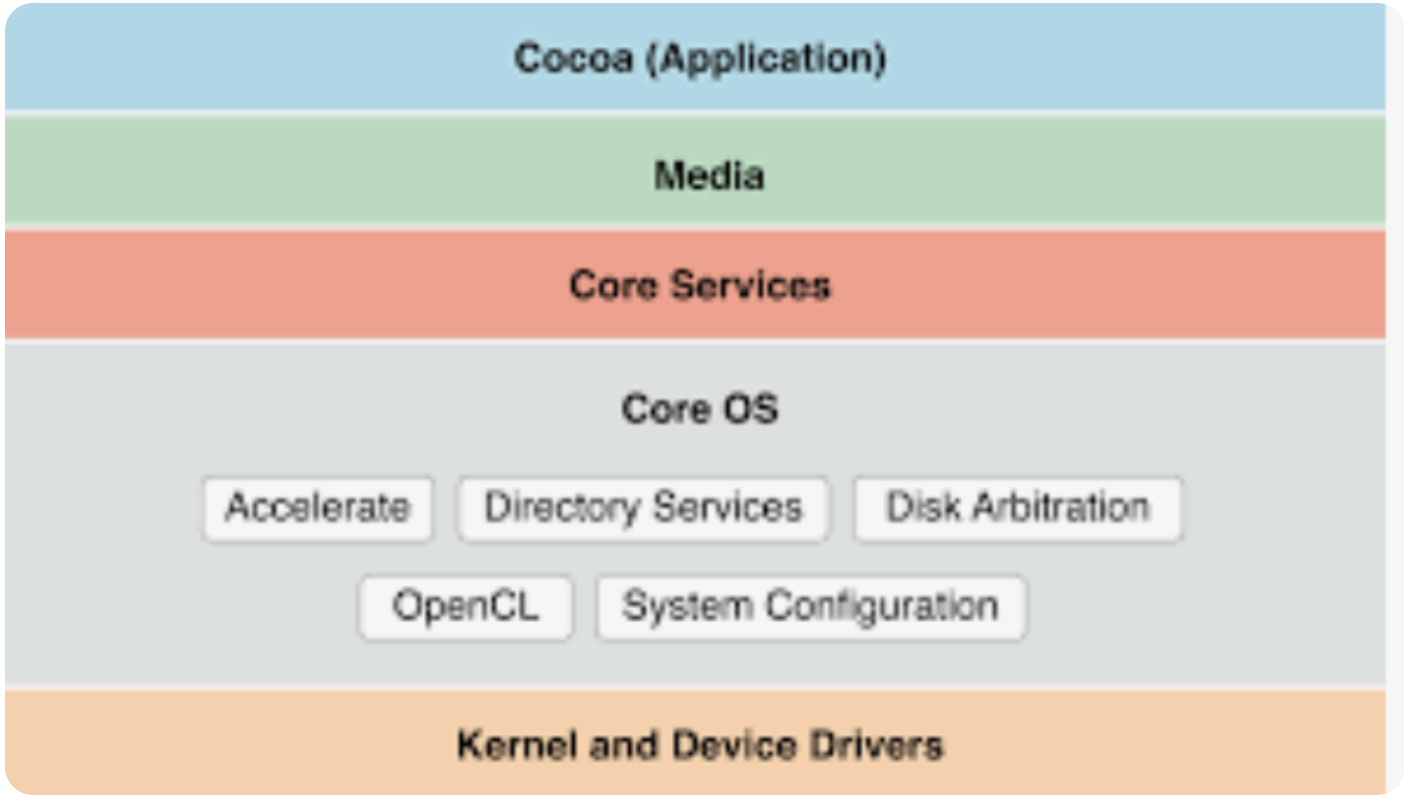
Darwin & the XNU Kernel

The foundation. Darwin is Apple's open-source OS, built on the **XNU kernel** (X is Not Unix).

- **XNU:** A hybrid kernel combining Mach microkernel with BSD components.
  - **Mach:** Process/thread management, virtual memory, IPC (via Mach ports).
  - **BSD:** File system (APFS, HFS+), networking (TCP/IP), security policies.
- **Security:** Mandatory Access Control (MAC) via **TrustedBSD**; code signing enforcement.
- **Drivers:** I/O Kit — an object-oriented C++ framework for device drivers.
- **Power Management:** Apple Silicon–specific power states, XPC for inter-process communication.

> **Note:** iOS does **not** use the standard macOS kernel exactly; it has a **restricted kernel** (no root access, no dynamic library loading after boot) specifically hardened for mobile.

Core OS & System Frameworks



Core OS & System Frameworks

The lowest-level APIs, built on top of Darwin. These are the **building blocks** that everything else depends on.

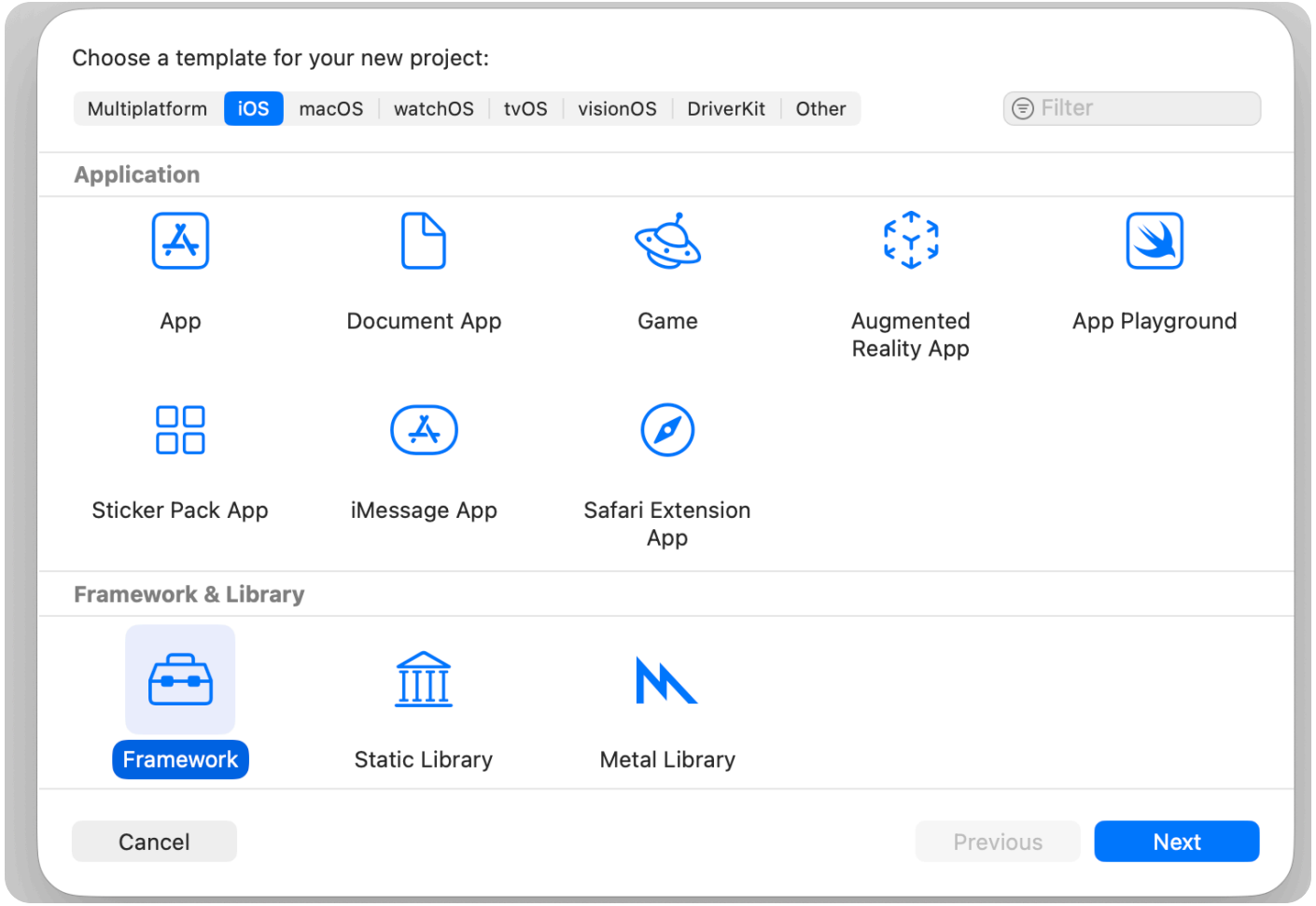
- **Security.framework:** Keychain Services, Certificate handling, CommonCrypto (AES, RSA, hashes).
- **System Configuration.framework:** Network reachability, DNS resolution.
- **libc / libobjc:** The C library and the Objective-C runtime.
- **Accelerate.framework:** Optimized vector/matrix math (DSP, BLAS, vImage) — critical for image/signal processing.
- **System Integrity: Code Signing** — every binary, framework, and app must be signed; the kernel refuses to execute unsigned code.

Core Services Layer

Provides fundamental services that support both apps and the frameworks above.

- **Core Foundation:** C-based API for strings, dates, collections, property lists.
- **Foundation:** The Objective-C/Swift counterpart to Core Foundation (NSString, NSArray, NSURL).
- **SQLite:** Embedded relational database (used internally by many Apple apps).
- **Grand Central Dispatch (GCD):** Queue-based concurrency; libdispatch handles thread pools.
- **Core Data:** Apple's ORM framework for persistent object graph management.
- **CloudKit / Core Location / Contacts:** High-level services for cloud sync, geolocation, and contacts.

Cocoa Touch Framework



Cocoa Touch Framework

The primary UI and interaction framework for iOS apps. **UIKit** is the backbone.

- **UIKit:** Views (UIView), controls (UIButton, UITableView), view controllers (UIViewController), gesture recognizers.
- **SwiftUI:** Declarative UI framework (iOS 13+) built on top of UIKit.
- **MapKit:** Maps and location visualization.
- **HealthKit / HomeKit / CarPlay:** Domain-specific frameworks.
- **Push Notifications:** APNs (Apple Push Notification Service) — a dedicated service outside the app process.
- **Multitasking:** `UIApplication` lifecycle hooks, background task scheduling (`BGTaskScheduler`).

Application Layer

Where the user interacts.

- **System Apps:** SpringBoard (Home Screen), Phone, Safari, Camera, Messages, App Store.
- **User Apps:** Apps developed with Xcode and deployed via App Store (or sideloaded with a developer account).
- **App Sandbox:** iOS's primary security boundary.
  - Each app gets a **container directory** (`~/Library/`, `~/Documents/`, `~/tmp/`).
  - App A **cannot** read or write App B's container.
  - **Entitlements** define additional capabilities (Keychain sharing, App Groups, push notifications, HealthKit).
  - **No arbitrary file system access** — apps cannot touch each other's data or system files.

Info.plist: The Control Center

`Info.plist` is the **metadata file** for your app. Without it, iOS doesn't know how to launch or configure your app.

Key Roles:

1. **Identity:** Bundle identifier (e.g., `edu.sharif.mp`), version strings.
2. **Launch Configuration:** `UILaunchStoryboardName`, `UIApplicationSceneManifest`.
3. **Capabilities:** Declares which features your app uses (Camera, Bluetooth, Background Modes).
4. **URL Schemes:** Defines custom URL schemes your app handles (e.g., `myscheme://`).
5. **Privacy Descriptions:** Required strings for accessing sensitive data (NSCameraUsageDescription, NSLocationWhenInUseUsageDescription).

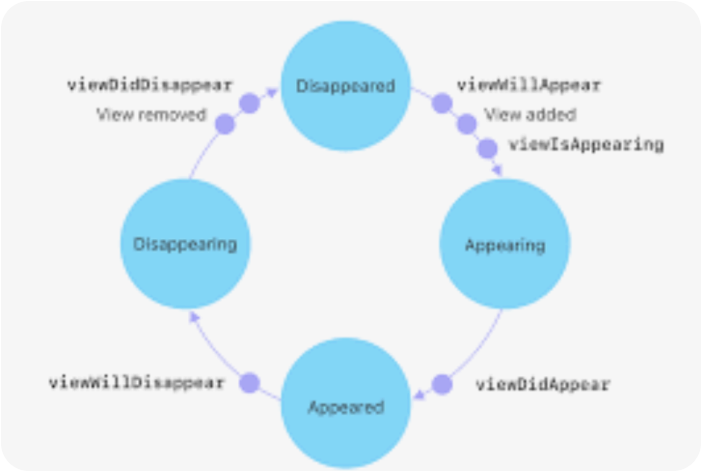
IPA Structure

An **IPA (iOS App Store Package)** is a **ZIP file** containing the app and its resources.

- `Info.plist` : The metadata bundle.
- `MyApp.app/` : The actual application bundle.
  - `MyApp` : The main executable (Mach-O binary).
  - `Frameworks/` : Embedded dynamic libraries.
  - `PlugIns/` : App extensions (Widgets, Share Extensions).
  - `Assets.xcassets/` : Images, colors, SF Symbols.
  - `Base.lproj/` : Storyboards and XIBs.
- `Payload/MyApp.app/_CodeSignature/CodeResources` : Code signing hash list.
- **No runtime code modification:** The entire bundle is signed; any modification invalidates the signature → app refuses to launch.

View Controller Lifecycle

The **View Controller Lifecycle** is the most critical concept. If you get this wrong, your UI won't update properly.



View Controller Lifecycle

Appearing States

- **loadView:** View is created programmatically.
- **viewDidLoad:** View loaded from storyboard/XIB.
- **viewWillAppear:** About to animate onto screen.
- **viewDidAppear:** Fully visible and interactable.

Disappearing States

- **viewWillDisappear:** About to leave screen.
- **viewDidDisappear:** No longer visible.

Deep Dive: Memory Warning & Suspension

Why did my app lose its state?

1. **System Memory Warning:** iOS notifies active apps via `didReceiveMemoryWarning`.
2. **App is Suspended:** iOS moves app to background; it stays in memory but is **frozen** (no CPU cycles).
3. **System needs memory:** iOS may **terminate** suspended apps without warning.
4. **User returns to App:** iOS **re-launches** from scratch; `viewDidLoad` runs again.
5. **Your Job:** Persist data in `UserDefaults`, `SQLite`, `Core Data`, or files so the UI can be restored.

> **Golden Rule:** Do **not** trust that your in-memory state survives a background termination.

Summary: System Thinking

- **Architecture:** Layered stack (XNU → Core OS → Core Services → Cocoa Touch → App).
- **Runtime:** Native machine code via AOT compilation (Swift) + Objective-C message dispatch.
- **Security:** Code signing + App Sandbox + Entitlements.
- **Lifecycle:** You do not control when your app dies. You only control how it *reacts* to lifecycle and memory events.

Mini Lab

System Behavior Inspector

Objective: Create a simple app that logs lifecycle events.

1. **Log Events:** Override `viewDidLoad`, `viewWillAppear`, `viewDidAppear`, `viewWillDisappear`, `viewDidDisappear`, `didReceiveMemoryWarning`.
2. **Visualize:** Print logs to the Xcode console.
3. **Simulate Termination:**
   - Run app on a real device (or Simulator).
   - Press Home.
   - In Xcode Debug Navigator, force terminate the process.
   - Relaunch and observe `viewDidLoad` running again.

Real-Life Application

Background Audio Player Logic

- **Requirement:** Music must continue when screen is off or app is backgrounded.
- **System Conflict:** Suspended apps cannot execute code.
- **Solution:** Enable **Background Modes** → **Audio** in Capabilities.
  - `AVAudioSession` configures audio category (`playback`).
  - iOS keeps the audio process alive even when suspended.
  - Register for remote control events (Control Center, Lock Screen).

```
// Swift implementation
AVAudioSession.sharedInstance().setCategory(.playback, mode: .default)
AVAudioSession.sharedInstance().setActive(true)
```

Thank You! 🙌

Questions?

"A Swift developer walks into a bar.
The bartender says, 'Sorry, we don't serve your thread here.'
The developer replies, 'It's okay, I'll just dispatch_async to the main queue.'"