**IOS Architecture**

Apple describes the set of frameworks and technologies that are currently implemented within the iOS operating system as a series of layers. Each of these layers is made up of a variety of different frameworks that can be used and incorporated into your applications.The iOS Software Development Kit (SDK) contains the tools and interfaces needed to develop, install, run, and test native apps that appear on an iOS device’s Home screen. Native apps are built using the iOS system frameworks and Objective-C language and run directly on iOS. Unlike web apps, native apps are installed physically on a device and are therefore always available to the user, even when the device is in Airplane mode. They reside next to other system apps, and both the app and any user data is synced to the user’s computer through iTunes.

|  |
| --- |
| **Cocoa Touch** |
| **Media** |
| **Core Service** |
| **Core OS** |

**Cocoa Touch Layer**

Cocoa Touch Layer contains the key frame work for building ios apps. It also provides basic app infrastructure such as multitasking, push notification, and touch based input and many high level services for the app.

The following are the key technologies available in the cocoa touch layer.

* **App Extension:** An app extension lets you extend custom functionality and content beyond your app and make it available to users while they’re using other apps or the system. You create an app extension to enable a specific task; after users get your extension, they can use it to perform that task in a variety of contexts.

iOS supports app extensions for the following areas, which are known as extension points:

* [Share](https://developer.apple.com/library/ios/documentation/General/Conceptual/ExtensibilityPG/ShareSheet.html#//apple_ref/doc/uid/TP40014214-CH12-SW1) - an extension that enables your app to share content with users on social networks and other sharing services.
* [Action](https://developer.apple.com/library/ios/documentation/General/Conceptual/ExtensibilityPG/Services.html#//apple_ref/doc/uid/TP40014214-CH13-SW1) - an extension which allows creating custom action buttons in the Action sheet to let users view or transform content originating in a host app.
* [Photo Editing](https://developer.apple.com/library/ios/documentation/General/Conceptual/ExtensibilityPG/Photos.html#//apple_ref/doc/uid/TP40014214-CH17-SW1) - an extension that lets users edit a photo or a video within the Photos app.
* [Document Provider](https://developer.apple.com/library/ios/documentation/General/Conceptual/ExtensibilityPG/FileProvider.html#//apple_ref/doc/uid/TP40014214-CH18-SW1) - an extension used for allowing other apps to access the documents managed by your app.
* [Custom Keyboard](https://developer.apple.com/library/ios/documentation/General/Conceptual/ExtensibilityPG/Keyboard.html#//apple_ref/doc/uid/TP40014214-CH16-SW1) - an extension that replaces the system keyboard.
* **Handoff:** Handoff lets users start an activity on one device and seamlessly resume the activity on another device. Provide continuity for users with multiple devices by supporting Handoff in your apps and websites. For example, a user who is browsing a long article in Safari moves to an iOS device that's signed into the same Apple ID, and the same webpage automatically opens in Safari on iOS, with the same scroll position as on the original device. Handoff makes this experience as seamless as possible.
* **Document Picker:** The document picker feature lets users select documents from outside your app’s sandbox. These include documents stored in iCloud Drive and documents provided by a third-party extension. Users can open these documents directly, editing them in place. This access simplifies sharing documents between apps and enables more complex workflows. For example, users can easily edit a single document using multiple apps.
* **AirDrop:** AirDrop lets users share photos, documents, URLs, and other kinds of data with nearby devices. Support for sending files to other iOS devices using AirDrop is built into the existing [UIActivityViewController](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIActivityViewController_Class/index.html#//apple_ref/occ/cl/UIActivityViewController) class. This class displays different options for sharing the content that you specify. If you are not yet using this class, you should consider adding it to your interface.
* **TextKit:** Text Kit is a set of classes and protocols in the UIKit framework providing high-quality typographical services that enable apps to store, lay out, and display text with all the characteristics of fine typesetting, such as kerning, ligatures, line breaking, and justification. Text Kit is built on top of Core Text, so it provides the same speed and power. UITextView is fully integrated with Text Kit; it provides editing and display capabilities that enable users to input text, specify formatting attributes, and view the results. TextKit is a full-featured, high-level set of classes for handling text and fine typography. Using TextKit, you can lay out styled text into paragraphs, columns, and pages; you can flow text around arbitrary regions such as graphics; and you can use it to manage multiple fonts. If you were considering using Core Text to implement text rendering, you should consider TextKit instead. TextKit is integrated with all UIKit text-based controls to enable apps to create, edit, display, and store text more easily—and with less code than was previously possible in iOS.
* **UIKit Dynamics:** It is a full physics engine integrated into UIKit. It allows you to create interfaces that feel real by adding behaviors such as gravity, attachments (springs) and forces. You define the physical traits that you would like your interface elements to adopt, and the dynamics engine takes care of the rest.

UIKit dynamics supports the following types of behaviors:

* **UIGravityBehavior** – Provides gravitational behavior to your views.
* **UICollisionBehavior** – Provides collision detection.
* **UISnapBehavior** – Causes a view to move to and snap to a specific point on the interface as if connected to it.
* **UIAttachmentBehavior** – Specifies a dynamic connection between two dynamic items, or between a dynamic item and an anchor point.
* **UIPushBehavior** – Applies a continuous or instantaneous force to one or more dynamic items, causing those items to change position accordingly.

### Multitasking: Battery life is an important consideration for users of iOS devices and the multitasking model in iOS is designed to maximize battery life while giving apps the time they need to do critical work. When the user presses the Home button, the foreground app shifts to a background execution context. If the app has no more work to do, it is suspended from active execution and put into a "freeze-dried” state, where it remains in memory but does not execute any code. Apps that do need specific types of work can ask the system for background execution time.

**Cocoa Touch Frame Work:**

The following sections describe the frameworks of the Cocoa Touch layer and the services they offer.

**Address Book UI Framework:**

* Address Book UI is an iOS framework for displaying, selecting, editing, and creating contacts in a user’s Address Book. Similar to the Message UI framework, Address Book UI contains a number of controllers that can be presented modally, to provide common system functionality in a uniform interface.
* To use the framework, add both AddressBook.framework and AddressBookUI.framework to our project,under the “Link Binary With Libraries” phase.
* The Address Book technology for iOS provides a way to store people’s contact information and other personal information in a centralized database, and to share this information between applications. The technology has several parts:
* The Address Book framework provides access to the contact information.
* The Address Book UI framework provides the user interface to display the information.
* The Address Book database stores the information.
* The Contacts application provides a way for users to access their contact information.

**EventKit UI Framework:**

* To work with reminder and calendar events, we need to link against **EventKit**. We will also need a persistent store to save reminder items. Conveniently, EventKit provides this : **EKEventStore**. An **EKEventStore** allows us to fetch, create, edit, and delete events from a user’s Calendar database.
* Both reminders and calendar data are stored in the Calendar database. Ideally, we will have only one event store for your entire app, and you will instantiate it once. That’s because an **EKEventStore** object requires a relatively large amount of time to initialize and release.
* The Event Kit UI framework provides the classes needed to create, edit, and display events using a view controller. It provides several configurable view controller classes.

**GameKit Framework:**

Game Kit provides three separate pieces of functionality:

* Game Center offers a centralized game service that connects players to each other. Game Center implements many different features:
* Friends allow players to create anonymous online personas. Users connect to Game Center and interact with other players through analias. Players can set status messages as well as mark other players as friends.
* Multiplayer allows your game to create network matches that connect players through Game Center. Players can invite their friends or be connected to anonymous players. Most importantly, players can receive invitations to join a match even when your game is not running. Your game is running on each device and the instances of your game exchange match and voice data with each other.
* Turn-Based Gaming provides store-and-forward network match infrastructure where the match is played out over a series of discrete turns. This kind of match can be played without requiring all of the players to be connected to Game Center simultaneously.
* Leaderboards allow your game to store and fetch player scores from Game Center.
* Achievements provide the ability to track a player’s accomplishments in your game.
* Challenges allow a player to challenge other players to complete an achievement or to beat a leaderboard score.

**iAd Framework:**

* iAd allows our application to earn revenue by displaying advertisements to the user. Our application dedicates a portion of its user interface to display advertisements and in turn us to receive revenue when users view or click those advertisements.
* While we are developing our application, iAd sends test advertisements to help us to verify our implementation is correct. To receive live advertisements from iAd in a release application, we need to integrate the iAd Framework in our application and submit our binary using iTunes Connect.

**MapKit Framework:**

* MapKit is a really neat API available on the iPhone that makes it easy to display maps, jump to coordinates, plot locations, and even draw routes and other shapes on top
* The Map Kit framework provides an interface for embedding maps directly into your own windows and views. This framework also provides support for annotating the map, adding overlays, and performing reverse-geocoding lookups to determine placemark information for a given map coordinate.

**Message UI Framework:**

* The Message UI framework provides specialized view controllers for presenting standard composition interfaces for email and SMS (Short Messaging Service) text messages. Use these interfaces to add message delivery capabilities without requiring the user to leave the app.
* To display a composition interface, present the corresponding view controller modally from our app. Once presented, the user has the option to customize the contents before sending or canceling the message.

**Notification Center Framework:**

* The Notification Center framework helps us to create and manage extensions—typically called widgets—in the Today view. The framework provides API we can use to specify whether a widget has content to display and to customize aspects of its appearance and behavior on both platforms.
* In OS X, the Notification Center framework provides ways to customize the editing and searching experience in a widget.

**PushKit Framework:**

* The PushKit framework (PushKit.framework) provides registration support for VoIP apps. This framework replaces the previous APIs for registering VoIP apps. Instead of keeping a persistent connection open, and thus draining the device’s battery, an app can use this framework to receive push notifications when there is an incoming call.

**UIKit Framework:**

* The UIKit framework (UIKit.framework) provides the crucial infrastructure needed to construct and manage iOS apps. This framework provides the window and view architecture needed to manage an app’s user interface, the event handling infrastructure needed to respond to user input, and the app model needed to drive the main run loop and interact with the system.