**Layers of IOS Architecture**

**1] Cocoa Touch layer**

* This layer consists of frameworks help to build the app and defines the appearance of the app.
* It provides the basic app infrastructure and high level services like multitasking, push notification etc.

**High level features**

**i) app extension**

* This lets the user to extend selected areas of a system.
* It is a code enables custom functionality within the context of the user task.
* The extension can be enabled by the user in the setting app.

IOS supports the app extensions for the following areas, which are called as extension points:

* Share : Share content with the social website or other entities.
* Action : Perform a task with current content.
* Widget : Provide a quick update.
* photo editing : edit the photos or videos using editing app.
* Document provider : Provide a document storage location which can be accessed by other app.
* Custom keyboard : Provides the custom keyboard to the user to use the app in place of the system keyboard.

**ii) Handoff**

* This enables the user to begin a task on one device and switch to other device, resume the same activity on it.
* For eg: user browsing an article on safari can continue the browsing on an iOS device, if the two devices have same Apple ID. The same page automatically opens on the iOS device.
* When the user begins a task user object is created which contains the data to resume an activity on other device.
* When the user chooses to resume the same activity, the object is sent to the resuming device.

**iii) Document Picker**

* Document picker view controller ([UIDocumentPickerViewController](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIDocumentPickerViewController_Class/index.html" \l "//apple_ref/occ/cl/UIDocumentPickerViewController" \t "_self)) grants the user access to the files outside the sandbox.
* It helps to share files between two apps and also allows the user to edit the file using more than one app.
* For eg: the iCloud document provider allows to access the files inside app's iCloud container.

**iv) Air Drop**

* It helps to share photos, files, links etc with nearby devices.
* This is implimented in the class [UIActivityViewController](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIActivityViewController_Class/index.html#//apple_ref/occ/cl/UIActivityViewController).

To recieve files sent using AirDrop, the app must do the following:

* Declare support for the appropriate files in Xcode.
* Implement the [application:openURL:sourceApplication:annotation:](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplicationDelegate_Protocol/index.html" \l "//apple_ref/occ/intfm/UIApplicationDelegate/application:openURL:sourceApplication:annotation:" \t "_self) method in your app delegate.

**v) Text Kit**

* It consists of high level set of classes for handling text and toporaphy.
* Using Text Kit we can create a paragraphs, columns and pages of styled text.
* Text Kit is integrated with all UI text based controls to enable apps to create, edit, display and store the text more easily.

TextKit comprises of new UIkit classes along with extension to existing classes, they are:

* The [NSAttributedString](https://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSAttributedString_Class/index.html" \l "//apple_ref/occ/cl/NSAttributedString" \t "_self) class has been extended to support new attributes.
* The [NSLayoutManager](https://developer.apple.com/library/ios/documentation/UIKit/Reference/NSLayoutManager_Class_TextKit/index.html" \l "//apple_ref/occ/cl/NSLayoutManager" \t "_self) class generates glyphs and lays out text.
* The [NSTextContainer](https://developer.apple.com/library/ios/documentation/UIKit/Reference/NSTextContainer_Class_TextKit/index.html" \l "//apple_ref/occ/cl/NSTextContainer" \t "_self) class defines a region where text is laid out.
* The [NSTextStorage](https://developer.apple.com/library/ios/documentation/UIKit/Reference/NSTextStorage_Class_TextKit/index.html" \l "//apple_ref/occ/cl/NSTextStorage" \t "_self) class defines the fundamental interface for managing text-based content.

**vi) UIKit Dynamics**

* The apps can specify dynamic behaviours for UIView objects and for other objects that conforms to the UIDynamicItem protocol.
* It helps to improve the user experience of the app by incorporating real world behavior and characteristics into app's UI.
* A [UIAttachmentBehavior](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIAttachmentBehavior_Class/index.html" \l "//apple_ref/occ/cl/UIAttachmentBehavior" \t "_self) object specifies a connection between two dynamic items or between an item and a point. When one item moves the attached item also moves.
* A [UICollisionBehavior](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UICollisionBehavior_Class/index.html" \l "//apple_ref/occ/cl/UICollisionBehavior" \t "_self) object lets dynamic items participate in collisions with each other and with the behavior’s specified boundaries.
* A [UIGravityBehavior](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIGravityBehavior_Class/index.html" \l "//apple_ref/occ/cl/UIGravityBehavior" \t "_self) object specifies a gravity vector for its dynamic items.
* A [UIPushBehavior](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIPushBehavior_Class/index.html" \l "//apple_ref/occ/cl/UIPushBehavior" \t "_self) object specifies a continuous or instantaneous force vector for its dynamic items.
* A [UISnapBehavior](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UISnapBehavior_Class/index.html" \l "//apple_ref/occ/cl/UISnapBehavior" \t "_self) object specifies a snap point for a dynamic item.

**vii) Multitasking**

* Battery life is an important consideration in iOS devices, multitasking is mainly done to increase the battery life.
* When the Home button is pressed the foreground app shifts to background execution context.
* When the app is suspended from execution, it goes to the state called freeze dried, where it remains in memory but does not execute.
* Apps which needs a specific work can ask the system for background execution time :
* App can request for finite time to execute some important task.
* Apps which supports specific services can like audio playback can request for time.
* An app can download content periodically from the network.
* An app can download content in response to a push notification.

**viii) Auto Layout**

* Using auto layout, we can define rules for how to lay out the elements in User interface.
* Entities used in auto layout provides the following benefits: support localization through the swapping of strings, support mirroring of user interface elements for right-to-left languages, such as Hebrew and Arabic, promote a better separation of responsibilities between objects in the view and controller layers.

**ix) Storyboards**

* It is a way of designing the app's user interface.
* We can create the user interface at one place and also can see the view, view controller and also understand how they work together.
* The important ability of storyboard is segue, which is transition from one view controller to other. These transitions we can observe in Xcode.
* We can use single storyboard or multiple view storyboard to organize the portions of interface. But during Build time Xcode takes the contents of storyboard and divides it into discrete pieces and that can be loaded individually for better performance.
* App directly never access these pieces of content but UIKit framework provides classes to access these contents.

**x) UI State preservation**

* It provides seamless experience to the users by having the app to be always running, even when it was not.
* When memory pressure occurs the foreground app may terminate and go to background by preserving some of the information to restore its view and view controller.

**xi) Apple Push notification Service**

* It alerts the users about new notifications, even when the app is not actively running.
* User can push text notifications, add a badge to an app icon, or trigger audible alerts on user devices at any time using this service.
* This tells the user to open the app to get information regarding updates.
* External server generates the push notification.

**xii) Local Notifications**

* Here the notifications are generated locally, instead of lying on external server.
* Apps running in background uses the local notification to get user attention when important even happen.
* The local notifications are independent of the user app. After a notification is scheduled, the system manages the delivery of it.

**xiii) Gesture Recognizers**

* Gesture recognizers detect common types of gestures, like swipes and pinches in user app’s views.
* They use the same heuristics as the system for detecting gestures, gesture recognizers offer a consistent behavior for your apps.
* All gesture recognizers are based on the [UIGestureRecognizer](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIGestureRecognizer_Class/index.html" \l "//apple_ref/occ/cl/UIGestureRecognizer" \t "_self) class, which defines the basic behavior.

**Cocoa touch frameworks:**

**i) Address Book UI Framework:**

* This is a framework which displays the standard system interface for creating new contacts and edinting wxisting contacts.
* It helps to display the infromation regarding the contacts in the user app.

**ii) EventKit UI Framework:**

* It provides the view controller for presenting standard system interface for displaying and editing the events related to the calender.
* It is built upon event related data.

**iii) GameKit Framework:**

* It implements support for game center, which lets the user to share information about the game online.
* Game center provides support for the following features:
* Aliases, lets the user to create their own persona. Users can interact with other players using the aliases.
* Leader boards, lets the user app to post the score at game center and use it later.
* Matchmaking, lets to create multiplayers games by connecting players who logged in.
* Achievements, lets to record the score of a player who played the game.
* Challenges, lets the player challenge his friend.

**iv) iAD Framework:**

* It lets the user to deliver banner based advertisements from his app.
* Advertisements are incorporated into standard views that you integrate into your user interface and present when you want.
* The views themselves work with iAD service automatically handle all the works associated with loading and presenting rich media adds etc.

**v) MapKit Framework:**

* It helps to provide scrollable map in the app user interface.
* We can customize the map content and appearance.
* We can use custom overlays to intersperse our own content with the map content.
* It is integrated with Maps app and Apple's map server to facilitate the directions.
* Apps can also request walking and driving directions.

**vi) Message UI Framework:**

* It provide support for composing the messages and mails in the app.
* The composition support consists of a view controller interface that user has presented in his app.
* We can populate the fields of this view controller to set the recipients, subject, body content, and any attachments we want to include with the message.

**vii) Notification Center Framework:**

* It provides support for creating widgets that display information in Notification Center.

**viii) PushKit Framework:**

* It 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.

**ix) Twitter Framework:**

* This framework is replaced by social framework.
* It supports a UI for generating tweets and support for creating URLs to access the Twitter service.

**x) UIKit Framework:**

* It provides crucial infrastructure for implementing graphical, event-driven apps in iOS.
* The features are as follows:
* Basic app management and infrastructure
* User interface management
* A view controller model to encapsulate the contents of user interface.
* Support for handling touch and motion-based events.
* Support for a document model that includes iCloud integrations.
* Multitasking support.
* Printing Support.
* Cut, copy and paste support etc.