**Media Layer**

* The media layer consists of graphics, audio, video technologies needed in the app.

**Graphics Technology**

* Every app needs high quality graphics, iOS provides many technologies to support such a feature.
* We can use the standard views to deliver high quality graphics quickly on the screen or we can also create our own custom views.
* The following technologies can be used to deliver a good graphical experience:

**i) UIKit Graphics:** It defines high level support for drawing images and for animating content of the views. It provides fast and efficient way to render images and text based contents.

**ii) Core Graphics Framework:** It is a native drawing engine for iOS apps and provides support for custom 2D vector and image based redering.

**iii) Core Animation:** It is a foundational technology that optimizes the animation of the app.

**iv) Core image:** It provides support for manipulating videos and still images in a non destructive manner.

**v) Open GL ES and GLKit:** It handles 2D 3D rendering using hardware-accelerated interfaces. It is mainly use by game developers for smooth animations.

**vi) Metal:** It provides low overhead access to the A7 GPU, enabling incredibly high performance for the sophisticated graphics rendering and computation tasks. It eliminates performance bottlenecks, such as costly state validation.

**vii) Text Kit and Core Text:** It is a family of UIKit mainly used for typography and text management. Core Text is mainly used for typography and layout.

**viii) Image I/O:** It provides interfaces for reading and writing most image formats.

**ix) Photos Library:** Photos and PhotosUI frameworks provide access to user's photos, videos and media. It is used in places where you want to integrate the user’s own content with your app.

**Audio Technologies**

* iOS Audio technology works with underlying hardware to give rich audio experience for the users.
* This experience includes the ability to play and record the high quality audio.
* There are some audio technologies used in iOS:

**i) Media Player framework:** It provides easy access to iTunes to play the tracks in playlist.

**ii) AV Foundation :** It is an Objective C interface which supports recording and playback the audio and video.

**iii) OpenAL :** It is an industry standard technology for delivering positional audio. Game developers frequently use this technology to deliver high-quality audio using a set of cross-platform interfaces.

**iv) Core Audio:** It provides both simple and sophisticated interfaces for the recording and playback of audio and MIDI content. This framework is for advanced developers who need fine-grained control over their audio.

* iOS supports many industry standard and Apple specific audio format, includes the following:
* AAC
* Apple Lossless
* A-law
* IMA/ADPCM
* Linear PCM
* µ-law
* DVI/Intel IMA ADPCM
* Microsoft GSM 6.10
* AES3-2003

**Video Technologies:**

* It supports for managing static video content in your app or playing back streaming content from the Internet.
* The technologies supports the video recording and playback are:

**i)** [**UIImagePickerController**](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIImagePickerController_Class/index.html#//apple_ref/occ/cl/UIImagePickerController)**:** This class is a UIKit view controller for choosing user media files.

**ii) AVKit:** It provides simple-to-use interfaces for presenting videos. This framework supports both full-screen and partial-screen video playback and supports optional playback controls for the user.

**iii) AV Foundation:** It provides a advanced video playback and recording capabilities. This framework is used where more control over the presentation or recording of video is needed.

**iv) Core media:** It defines the low-level data types and interfaces for manipulating media. Most apps do not need to use this framework directly, but it is available when you need unparalleled control over your app’s video content.

**AirPlay**

* It lets apps audio video content to Apple TV and stream audio content to third party Air  
  Play speakers and receivers.
* It supports UIKit framework, Media Player framework, AV Foundation framework, and the Core Audio family of frameworks etc.
* Any content which is played using these frameworks is automatically made eligible for AirPlay distribution.
* Additional options for delivering content over AirPlay include the following:
* To extend the content displayed by an iOS device, create a second window object and assign it to any [UIScreen](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIScreen_Class/index.html" \l "//apple_ref/occ/cl/UIScreen" \t "_self) objects that are connected to the device through AirPlay.
* You can display Now Playing content on a connected Apple TV using AirPlay.
* Use the [AVPlayer](https://developer.apple.com/library/ios/documentation/AVFoundation/Reference/AVPlayer_Class/index.html" \l "//apple_ref/occ/cl/AVPlayer" \t "_self) class in AV Foundation to manage your app’s audio and video content. This class supports streaming its content via AirPlay when enabled by the user.
* For web-based audio and video, you can allow that content to be played over AirPlay by including an embed tag with the airplay attribute.

**Media Layer Frameworks**

The following describes the frameworks and their services:

**i) Assets Library Framework:**

* It provides access to the photos and videos managed in the user device.
* To access items in the user’s saved photos album or in any albums imported onto the device, this framework is used.
* New photos and videos can be saved to the album.

**ii) AV Foundation Framework:**

* It provides a set of Objective-C classes for playing, recording, and managing audio and video content.
* Use this framework when you want to integrate media capabilities seamlessly into your app’s user interface.
* Services offered by this framework are:
* Audio session management.
* App's media assets management.
* Support for editing media content.
* The ability to capture audio and video.
* The ability to playback audio and video.
* Track management.
* Track management.
* Metadata management. etc

**iii) AVKit Framework:**

* The AVKit framework (AVKit.framework) leverages existing objects in AV Foundation to manage the presentation of video on a device.
* It is intended as a replacement for the Media Player framework when you need to display video content.

**iv) Core Audio:**

* It is a frame work which provides native support for handling audio.
* These frameworks support the generation, recording, mixing, and playing of audio in your apps.
* Core Audio frameworks are:
* CoreAudio.framework: Defines the audio data types used throughout Core Audio.
* AudioToolbox.framework: Provides playback and recording services for audio files and streams. This framework also provides support for managing audio files, playing system alert sounds, and triggering the vibrate capability on some devices
* AudioUnit.framework: Provides services for using the built-in audio units, which are audio processing modules.
* CoreMIDI.framework: Provides a standard way to communicate with MIDI devices, including hardware keyboards and synthesizers.
* MediaToolbox.framework: Provides access to the audio tap interfaces.

**v) CoreAudioKit Framework:**

* provides standard views for managing connections between apps that support inter-app audio.
* One view provides a switcher that displays the icons of other connected apps and the other view displays the transport controls that the user can use to manipulate the audio provided by the host app.

**vi) Core Graphics Framework:**

* contains the interfaces for the Quartz 2D drawing API.
* Quartz is the same advanced, vector-based drawing engine that is used in OS X.
* It supports path-based drawing, ant aliased rendering, gradients, images, colors, coordinate-space transformations, and PDF document creation, display, and parsing.

**vii) Core Image Framework:**

* provides a powerful set of built-in filters for manipulating video and still images.
* For everything from touching up and correcting photos to face, feature, and QR code detection the built-in filters can be used.

**vii) Core Text Framework:**

* It offers a simple, high-performance C-based interface for laying out text and handling fonts.
* This framework is for apps that do not use TextKit but that still want the kind of advanced text handling capabilities found in word processor apps.
* It provides a sophisticated text layout engine, including the ability to wrap text around other content.

**viii) Core Video Framework:**

* It provides buffer and buffer-pool support for the Core Media framework.

**ix) Game controller Framework:**

* It lets the user to discover and configure Made-for-iPhone/iPod/iPad (MFi) game controller hardware in your app.
* Game controllers can be devices connected physically to an iOS device or connected wirelessly over Bluetooth.

**x) GLKit framework:**

* It contains a set of Objective-C based utility classes that simplify the effort required to create an OpenGL ES app.
* GLKit supports four key areas of app development:
* The [GLKView](https://developer.apple.com/library/ios/documentation/GLkit/Reference/GLKView_ClassReference/index.html" \l "//apple_ref/occ/cl/GLKView" \t "_self) and [GLKViewController](https://developer.apple.com/library/ios/documentation/GLkit/Reference/GLKViewController_ClassRef/index.html" \l "//apple_ref/occ/cl/GLKViewController" \t "_self) classes provide a standard implementation of an OpenGL ES–enabled view and associated rendering loop.
* The [GLKTextureLoader](https://developer.apple.com/library/ios/documentation/GLkit/Reference/GLKTextureLoader_ClassRef/index.html" \l "//apple_ref/occ/cl/GLKTextureLoader" \t "_self) class provides image conversion and loading routines to your app, allowing it to automatically load texture images into your context.
* The GLKit framework provides implementations of vectors, matrices, and quaternions, as well as a matrix stack operation that provides the same functionality found in OpenGL ES 1.1.
* The [GLKBaseEffect](https://developer.apple.com/library/ios/documentation/GLkit/Reference/GLKBaseEffect_ClassRef/index.html" \l "//apple_ref/occ/cl/GLKBaseEffect" \t "_self), [GLKSkyboxEffect](https://developer.apple.com/library/ios/documentation/GLkit/Reference/GLKSkyboxEffect_ClassRef/index.html" \l "//apple_ref/occ/cl/GLKSkyboxEffect" \t "_self), and [GLKReflectionMapEffect](https://developer.apple.com/library/ios/documentation/GLkit/Reference/GLKReflectionEffect_ClassRef/index.html" \l "//apple_ref/occ/cl/GLKReflectionMapEffect" \t "_self) classes provide existing, configurable graphics shaders that implement commonly used graphics operations.

**xi) Image I/O Framework:**

* It provides interfaces for importing and exporting image data and image metadata.
* This framework makes use of the Core Graphics data types and functions and supports all of the standard image types available in iOS.

**xii) Media Accessibility Framework:**

* It manages the presentation of closed-caption content in your media files.
* This framework works in conjunction with new settings that let the user enable the display of closed captions.

**xiii) Media Player Framework:**

* It provides high-level support for playing audio and video content from your app.
* We can use it as follows:
* Play video to a user’s screen or to another device over AirPlay.
* Access the user’s iTunes music library.
* Configure an manage movie play back.
* Display Now Playing information in the screen lock and App switcher.
* Display when video is being streamed over Airplay.

**xiv) Photos UI Framework:**

* The Photos UI framework (PhotosUI.framework) lets the user create app extensions for editing image and video assets in the Photos app.

**xv) Quartz Core Framework:**

* The Quartz Core framework (QuartzCore.framework) contains the Core Animation interfaces.
* Core Animation is an advanced compositing technology that makes it easy to create view-based animations that are fast and efficient.