ChatGPT

Danh sách các lớp UIKit bắt đầu bằng “UI”

Dưới đây là danh sách **toàn bộ các class trong UIKit** có tiền tố `UI`, bao gồm cả những class ít gặp. Các class được phân nhóm theo chức năng để tiện tra cứu. Mỗi class liệt kê gồm tên class, mô tả chức năng ngắn gọn và (nếu có thông tin) phiên bản iOS đầu tiên hỗ trợ.

Cơ sở hạ tầng ứng dụng (App Infrastructure)

- **UIResponder** – Lớp cơ sở cho các đối tượng có khả năng *xử lý sự kiện* trong ứng dụng, như chạm màn hình, nhập liệu, thông báo,... Mọi thành phần tương tác trong UIKit đều kế thừa từ `UIResponder` [1]. (Có từ iOS 2.0)
- **UIApplication** – Đại diện cho ứng dụng iOS đang chạy, quản lý vòng lặp sự kiện chính và trạng thái tổng thể của app. Thường mỗi app chỉ có một đối tượng `UIApplication` để điều phối mọi hoạt động [2]. (Có từ iOS 2.0)
- **UIWindow** – *Cửa sổ* của ứng dụng, là container nền cho các view. Mỗi app thường có một `UIWindow` chính để hiển thị giao diện. Các view được thêm vào window sẽ được hiển thị trên màn hình thiết bị [3]. (Có từ iOS 2.0)
- **UIScreen** – Cung cấp thông tin về *màn hình* của thiết bị (kích thước, độ phân giải, scale, v.v.). Thông qua `UIScreen`, app biết được kích thước và đặc điểm màn hình để bố trí giao diện phù hợp [4] [5]. (Có từ iOS 2.0)
- **UIDevice** – Cung cấp thông tin về *thiết bị* hiện hành, ví dụ kiểu máy, tên, phiên bản hệ điều hành, hướng màn hình,... `UIDevice.current` cho phép lấy đối tượng thiết bị hiện tại [6]. (Có từ iOS 2.0)

Views và Controls cơ bản

- **UIView** – *Lớp cơ bản cho mọi thành phần giao diện*. `UIView` quản lý một vùng hình chữ nhật trên màn hình và chịu trách nhiệm vẽ nội dung hoặc chứa các subview khác. Mọi control hay view hiển thị trên iOS đều kế thừa từ `UIView` [7]. (Có từ iOS 2.0)
- **UILabel** – Một subclass của `UIView` dùng để *hiển thị văn bản tĩnh*. Cho phép đặt nội dung chữ, font, màu chữ, căn lề... thường dùng cho các nhãn mô tả hoặc tiêu đề [8]. (Có từ iOS 2.0)
- **UIImageView** – View chuyên dụng để *hiển thị hình ảnh*. Cho phép hiển thị một ảnh tĩnh hoặc chuỗi ảnh động (animation) bên trong nó [9]. (Có từ iOS 2.0)
- **UIControl** – Lớp *cơ sở cho các control tương tác* (như nút bấm, công tắc...). Kế thừa từ `UIView` và bổ sung các tính năng về tương tác người dùng (nhận các sự kiện touch, thay đổi trạng thái) [10]. (Có từ iOS 2.0)
- **UIButton** – *Nút bấm* tương tác cơ bản. Cho phép người dùng tap để thực hiện một hành động. `UIButton` có thể tùy chỉnh tiêu đề, hình ảnh, và có các kiểu mặc định như system, detail, info,... [11]. (Có từ iOS 2.0)
- **UISegmentedControl** – Control hiển thị *nhiều nút lựa chọn* dưới dạng phân đoạn nằm ngang. Người dùng có thể chọn một trong các segment (tương tự nhóm radio button) [12]. (Có từ iOS 2.0)
- **UISwitch** – *Nút gạt bật/tắt* (on/off switch). Cho phép bật hoặc tắt một trạng thái nào đó, thường dùng trong phần cài đặt [13]. (Có từ iOS 2.0)
- **UISlider** – Thanh trượt cho phép chọn một giá trị trong khoảng liên tục. Người dùng có thể kéo núm trượt để điều chỉnh giá trị (ví dụ chỉnh âm lượng) [14]. (Có từ iOS 2.0)

- **UIProgressView** – Thanh *tiến trình* hiển thị mức độ hoàn thành của một tác vụ dưới dạng thanh tiến độ (progress bar) [15]. (Có từ iOS 2.0)
- **UIActivityIndicatorView** – View hiển thị *vòng xoay tải* (spinner) dạng bánh răng quay, thường dùng để báo hiệu đang xử lý hoặc tải dữ liệu [16]. (Có từ iOS 2.0)
- **UIPageControl** – Control dạng *chỉ báo phân trang*, gồm các chấm tròn biểu thị số trang và trang hiện tại (thường dùng trong giao diện dạng slideshow hoặc onboarding) [17]. (Có từ iOS 2.0)
- **UIStepper** – Control gồm hai nút "+" và "–" cho phép tăng hoặc giảm một giá trị tuần tự (ví dụ chọn số lượng) [18]. (Có từ iOS 5.0)
- **UIScrollView** – View hỗ trợ *cuộn nội dung*. Nó bao bọc các view con và cho phép người dùng cuộn, kéo rê để xem nội dung vượt quá kích thước màn hình. `UIScrollView` cũng hỗ trợ zoom và phân trang nội dung [19]. (Có từ iOS 2.0)
- **UITableView** – *Bảng danh sách* cuộn được, hiển thị một cột các cell (hàng) có thể tái sử dụng. Thường dùng để hiển thị danh sách dữ liệu theo hàng dọc, có hỗ trợ chia section, header/footer cho nhóm nội dung [20]. (Có từ iOS 2.0)
- **UICollectionView** – View hiển thị danh sách các item với *bố cục linh hoạt*. Khác với UITableView (chỉ một cột dọc), `UICollectionView` cho phép bố trí nhiều cột, lưới, hoặc tùy biến layout (thông qua `UICollectionViewLayout`) [21]. (Có từ iOS 6.0)
- **UIWebView** – View nhúng *trình duyệt Web* bên trong ứng dụng, cho phép tải và hiển thị trang web hoặc HTML. Đã **ngừng dùng từ iOS 12.0** và được thay thế bởi `WKWebView` hiệu năng cao hơn [22] [23]. (Có từ iOS 2.0)
- **UIVisualEffectView** – View đặc biệt để áp dụng *hiệu ứng hình ảnh* mờ hoặc sáng lên nội dung. Thường dùng với các hiệu ứng như `UIBlurEffect` hay `UIVibrancyEffect` để tạo nền mờ kính (blur) hoặc làm nổi bật nội dung trên nền mờ [24]. (Có từ iOS 8.0)
- **UIBlurEffect** – Hiệu ứng mờ nền. Khi áp dụng một đối tượng `UIBlurEffect` vào `UIVisualEffectView`, nội dung phía sau view đó sẽ bị làm mờ đi theo kiểu blur được chọn [25]. (Có từ iOS 8.0)
- **UIVibrancyEffect** – Hiệu ứng tăng độ *tương phản* cho nội dung. Thường sử dụng cùng `UIBlurEffect` để giữ cho nội dung (như chữ/icon) trên nền mờ vẫn rõ ràng, dễ đọc [26]. (Có từ iOS 8.0)

Thanh công cụ và thanh điều hướng (Bars & Navigation UI)

- **UINavigationBar** – Thanh điều hướng nằm ở đầu màn hình, hiển thị tiêu đề trang hiện tại và các nút điều hướng (ví dụ nút "Back"). `UINavigationBar` là phần giao diện chính của navigation controller, thể hiện cấu trúc phân cấp của ứng dụng [27]. (Có từ iOS 2.0)
- **UINavigationItem** – Đại diện cho *nội dung trên thanh điều hướng* (title, nút bar button). Mỗi view controller được push vào `UINavigationController` thường có một `UINavigationItem` chứa tiêu đề và các nút cần hiển thị trên `UINavigationBar` [28]. (Có từ iOS 2.0)
- **UIToolbar** – Thanh công cụ thường nằm ở cạnh dưới hoặc trên giao diện, chứa các nút tác vụ. `UIToolbar` có thể chứa nhiều `UIBarButtonItem` để thực hiện các hành động nhanh. (Có từ iOS 2.0)
- **UITabBar** – Thanh tab nằm ở cuối màn hình, cho phép chuyển đổi giữa các *tab* (màn hình chức năng) trong app. Gồm một hàng các `UITabBarItem` (biểu tượng + nhãn) để người dùng chọn [29]. (Có từ iOS 2.0)
- **UITabBarItem** – Đại diện cho một *mục tab* trên `UITabBar`, chứa thông tin về tiêu đề, icon của tab. Mỗi view controller trong `UITabBarController` tương ứng với một `UITabBarItem` trên thanh tab [30]. (Có từ iOS 2.0)
- **UIBarButtonItem** – Đối tượng *nút trên thanh công cụ hoặc thanh điều hướng*. Khác với UIButton (nút trong view), `UIBarButtonItem` được dùng trong `UINavigationBar` hoặc `UIToolbar` để hiển thị nút hành động [31]. (Có từ iOS 2.0)

- **UIBarItem** – Lớp cơ sở cho các mục trên thanh (như `UIBarButtonItem`, `UITabBarItem`). Định nghĩa các thuộc tính chung như tiêu đề, icon, trạng thái bật/tắt cho những item nằm trong thanh công cụ hoặc thanh tab [32].
- **UISearchBar** – Thanh tìm kiếm giao diện người dùng, gồm trường nhập text và nút Search. Thường được dùng để người dùng nhập từ khóa tìm kiếm, có thể tích hợp trong thanh điều hướng hoặc hiển thị độc lập [33]. (Có từ iOS 3.0)

Layout và Giao diện (Layout & Interface)

- **UIStackView** – *Container sắp xếp các view theo hàng hoặc cột*. Đây là layout *định hướng tuyến tính* giúp tự động phân bố và co giãn subview bên trong một cách linh hoạt. `UIStackView` đơn giản hóa việc tạo giao diện đáp ứng mà không cần nhiều ràng buộc Auto Layout thủ công [34]. (Giới thiệu từ iOS 9.0 [35])
- **UILayoutGuide** – Đối tượng đại diện cho một vùng *hướng dẫn bố cục vô hình* dùng trong Auto Layout. `UILayoutGuide` cho phép tạo ra điểm neo hoặc vùng để các constraint liên kết, giúp bố trí khoảng trống hoặc căn lề một cách rõ ràng hơn [36]. (Có từ iOS 9.0)
- **UILayoutSupport** – Giao thức (được các lớp UIKit triển khai sẵn) đại diện cho các khoảng không gian cố định ở cạnh trên/dưới của màn hình (ví dụ Top/Bottom Layout Guide trước iOS 11). Được dùng trong Auto Layout để bố trí nội dung tránh vùng thanh trạng thái hoặc thanh tab [37]. (Có từ iOS 7.0, *đã được thay thế* bằng safe area từ iOS 11)
- **UITraitCollection** – Mô tả *các đặc điểm môi trường giao diện* (size class, giao diện tối/sáng, độ phân giải màn hình, kiểu thiết bị,...). `UITraitCollection` được dùng để xác định giao diện đang ở trạng thái nào (ví dụ iPhone hay iPad, xoay ngang hay dọc) để app tùy biến bố cục cho phù hợp [38]. (Có từ iOS 8.0)
- **UIAppearance** – Cơ chế *tùy biến giao diện tổng thể* cho các UIView/control. Thông qua `UIAppearance`, có thể thiết lập giao diện (màu sắc, font chữ,...) mặc định cho toàn bộ thành phần UI trên phạm vi toàn app (ví dụ đặt màu thanh navigation cho mọi `UINavigationBar`) [39]. (Có từ iOS 5.0)
- **UINib** – Đại diện cho một file nib (xib) của Interface Builder được *nạp sẵn vào bộ nhớ*. Sử dụng `UINib` giúp tải và tạo các đối tượng giao diện từ file XIB nhanh hơn khi dùng lặp lại nhiều lần (cache nội dung nib) [40]. (Có từ iOS 4.0)
- **UIStoryboard** – *Bảng phân cảnh giao diện* giới thiệu từ iOS 5, cho phép thiết kế nhiều màn hình và chuyển cảnh giữa chúng trong một file. Lớp `UIStoryboard` đại diện cho tệp storyboard, cung cấp khả năng khởi tạo view controller dựa trên `Storyboard ID` đã thiết lập [41]. (Có từ iOS 5.0)
- **UIStoryboardSegue** – Đối tượng biểu diễn một *chuyển tiếp (segue)* giữa hai màn hình trên storyboard. Khi một segue được kích hoạt (ví dụ nhấn nút chuyển màn hình), UIKit sẽ tạo `UIStoryboardSegue` để thực hiện việc chuyển từ source view controller sang destination view controller theo kiểu định sẵn (show, modal, popover...) [42]. (Có từ iOS 5.0)

View Controller và điều hướng (View Controllers & Navigation)

- **UIViewController** – Lớp cơ bản quản lý một *màn hình* hoặc một *nhóm view* trong ứng dụng MVC. Mỗi `UIViewController` quản lý vòng đời các view của nó, xử lý sự kiện giao diện và tương tác giữa *View* và *Model*. Đây là nền tảng của kiến trúc điều khiển giao diện trong iOS [43]. (Có từ iOS 2.0)
- **UINavigationController** – Một subclass của `UIViewController` dùng để *điều hướng phân cấp*. Nó quản lý một stack các view controller con (mỗi VC là một màn hình) và cung cấp cơ chế push/pop để đi tới hoặc quay lui giữa các màn hình. `UINavigationController` tự động cung cấp `UINavigationBar` để hiển thị tiêu đề và nút điều hướng [44]. (Có từ iOS 2.0)

- **UITabBarController** – View controller chứa một *giao diện tab* ở dưới cùng, cho phép chuyển đổi giữa các chế độ màn hình độc lập. Mỗi tab là một root view controller riêng, và `UITabBarController` quản lý một mảng các view controller này cùng với `UITabBar` tương ứng [45]. (Có từ iOS 2.0)
- **UISplitViewController** – View controller *đa panel*, thường dùng trên iPad, hiển thị hai view controller cạnh nhau (master-detail). Ví dụ: danh sách ở panel bên trái và chi tiết ở panel bên phải. Hỗ trợ tự động ẩn/hiện panel tùy theo kích cỡ màn hình (khi xoay ngang/dọc) [46]. (Có từ iOS 3.2 – giới thiệu cùng iPad)
- **UIPageViewController** – View controller cho phép *chuyển trang kiểu sách*. Nó quản lý một hoặc hai view controller con hiển thị cạnh nhau và cung cấp hiệu ứng lật trang hoặc cuộn trang. Người dùng có thể vuốt sang trái/phải để chuyển qua lại giữa các trang (màn hình) [47]. (Có từ iOS 5.0)
- **UITableViewController** – View controller được thiết kế sẵn để quản lý một bảng `UITableView`. Lớp này tự làm delegate và dataSource cho table view, đồng thời quản lý việc load dữ liệu, cập nhật giao diện bảng dễ dàng. Thường dùng cho danh sách tĩnh hoặc danh sách đơn giản [48]. (Có từ iOS 2.0)
- **UICollectionViewController** – Tương tự UITableViewController nhưng dành cho `UICollectionView`. Lớp này hỗ trợ quản lý một collection view (dạng lưới hoặc tùy biến layout) và tự thiết lập các delegate/dataSource cần thiết, giúp nhanh chóng xây dựng màn hình danh sách phức tạp [49]. (Có từ iOS 6.0)
- **UIAlertController** – View controller dùng để trình bày *thông báo cảnh báo hoặc lựa chọn hành động*. Thay thế cho `UIActionSheet` và `UIAlertView` từ iOS 8. Nó có hai kiểu: `.alert` (hộp thoại giữa màn hình với nút bấm) và `.actionSheet` (thanh lựa chọn từ dưới trồi lên). Bạn có thể thêm các `UIAlertAction` (nút) vào để xử lý tương tác người dùng [50]. (Có từ iOS 8.0)
- **UIAlertAction** – Đối tượng đại diện cho một *nút hành động* trong UIAlertController. Bao gồm tiêu đề nút, kiểu nút (mặc định, cancel, destructive) và closure xử lý khi nút được chọn. (Có từ iOS 8.0) [51]
- **UIActionSheet** – *Hộp thoại lựa chọn* xuất hiện từ cạnh dưới màn hình, chứa một loạt nút cho người dùng chọn (thường dùng cho các hành động liên quan đến nội dung hiện tại). Lớp này **đã bị khai tử** từ iOS 8; nên dùng `UIAlertController` với kiểu `.actionSheet` thay thế [52]. (Có từ iOS 2.0, ngừng từ iOS 8.0)
- **UIAlertView** – *Hộp thoại thông báo* kiểu cũ (trung tâm màn hình với các nút OK/Cancel...). Đã **ngừng dùng từ iOS 8** và được thay bằng `UIAlertController` kiểu `.alert` [53]. (Có từ iOS 2.0, ngừng từ iOS 8.0)
- **UIActivityViewController** – View controller chuẩn cung cấp *các hành động chia sẻ và dịch vụ hệ thống*. Khi present `UIActivityViewController`, người dùng có thể chọn chia sẻ nội dung qua các dịch vụ như AirDrop, mạng xã hội, tin nhắn SMS, email, lưu ảnh,... Lớp này giúp kết nối ứng dụng với các tiện ích hệ thống một cách dễ dàng [54]. (Có từ iOS 6.0)
- **UICloudSharingController** – View controller giao diện chuẩn để *chia sẻ dữ liệu iCloud (CloudKit)*. Cho phép thêm hoặc gỡ người trong chia sẻ dữ liệu đám mây. Xuất hiện khi ứng dụng hỗ trợ chia sẻ record CloudKit với nhiều người dùng [55]. (Có từ iOS 10.0)
- **UIImagePickerController** – Giao diện chuẩn của iOS để *chọn hình ảnh hoặc video* từ thư viện hoặc chụp mới bằng camera. Bạn có thể cấu hình để chọn ảnh, quay video, và nhận kết quả trong delegate `UIImagePickerControllerDelegate` [56]. (Có từ iOS 2.0)
- **UIVideoEditorController** – Giao diện cho phép *chỉnh sửa video* ngắn (cắt độ dài) ngay trong app. Cho phép chọn một video từ thư viện và cắt trim đoạn cần thiết. (Có từ iOS 3.1) [57]
- **UIDocumentPickerViewController** – View controller cho phép *chọn tệp tài liệu từ bộ nhớ hoặc iCloud Drive*. Người dùng có thể duyệt các nhà cung cấp tài liệu (Files app) để chọn file mở trong app, hoặc lưu file từ app ra. Xuất hiện đầu tiên trong iOS 8 cùng iCloud Drive [58].

- **UIDocumentBrowserViewController** – Giao diện *trình duyệt tài liệu* toàn màn hình (giống ứng dụng Files) giới thiệu từ iOS 11. Cho phép người dùng duyệt và quản lý các file ngay trong app với giao diện thống nhất của hệ điều hành [59]. (Có từ iOS 11.0)
- **UIDocumentInteractionController** – Hỗ trợ *xem trước và mở file* bằng các ứng dụng khác. Ví dụ cho phép mở một PDF trong app hoặc chia sẻ sang app khác. Lớp này hiển thị menu hành động phù hợp (mở bằng app nào, copy, in ấn, v.v.) cho file được chỉ định [60]. (Có từ iOS 3.2)
- **UIPopoverController** – Controller dạng popover (cửa sổ nổi cỡ nhỏ) trên iPad. Cho phép hiển thị nội dung trong một khung nhỏ nổi lên (ví dụ danh sách tùy chọn) tại vị trí chỉ định trên màn hình. **Đã deprecated từ iOS 9.0** (thay bằng cơ chế `UIPopoverPresentationController` cùng `UIAlertController` hoặc segues) [61]. (Có từ iOS 3.2, ngừng từ iOS 9.0)
- **UIPopoverPresentationController** – Lớp quản lý việc trình bày một view controller dưới dạng *popover* trên iPad hoặc khi dùng tính năng Adaptive Presentation. Nó điều khiển vị trí hiển thị của popover, mũi tên hướng, và tương tác khi popover xuất hiện (ví dụ chạm ngoài để đóng) [62]. (Có từ iOS 8.0)
- **UIPresentationController** – Lớp trừu tượng quản lý *việc trình bày view controller* (như modal presentation). Cho phép tùy biến giao diện trình bày (frame, hoạt cảnh) cho các kiểu modal custom. Từ iOS 8, Apple tách việc quản lý trình bày khỏi UIViewController và đóng gói trong lớp này để dễ tùy biến [63]. (Có từ iOS 8.0)
- **UISearchController** – Controller giới thiệu từ iOS 8 để quản lý *tìm kiếm* trong app. Nó kết hợp một `UISearchBar` và một nội dung kết quả tìm kiếm (có thể là một table view controller). Khi người dùng bắt đầu tìm, `UISearchController` sẽ hiển thị kết quả trong màn hình con, giúp tách logik tìm kiếm khỏi giao diện chính [64]. (Có từ iOS 8.0)
- **UISearchDisplayController** – Controller tìm kiếm kiểu cũ (trước iOS 8) dùng để quản lý search bar và bảng kết quả trong iOS 3-7. **Đã ngừng từ iOS 8** khi `UISearchController` ra đời [65]. (Có từ iOS 3.0, ngừng từ iOS 8.0)
- **UIPageControl** – *(Đề cập ở mục Controls ở trên)* Hiển thị các dấu trang (dots) biểu thị số trang và trang hiện tại trong giao diện phân trang [17].
- **UIStoryboardPopoverSegue** – Một subclass của `UIStoryboardSegue` dùng trên iPad để trình bày segue dưới dạng popover [66]. (Có từ iOS 5.0)

(Lưu ý: Nhiều lớp delegate như UINavigationControllerDelegate, UIViewControllerTransitioningDelegate,... không liệt kê ở đây vì chúng là protocol*, không phải class cụ thể.)*

Văn bản và Nhập liệu (Text & Input)

- **UITextField** – Ô *nhập văn bản một dòng*, thường dùng cho nhập thông tin ngắn (như trường username/password). `UITextField` cho phép hiển thị placeholder, văn bản đã nhập, và tương tác bàn phím. Có thể tùy chỉnh kiểu bàn phím (số, email, v.v.) và có delegate để xử lý sự kiện như bắt đầu/kết thúc chỉnh sửa [67]. (Có từ iOS 2.0)
- **UITextView** – *Vùng văn bản đa dòng*, cho phép hiển thị và chỉnh sửa văn bản dạng dài. `UITextView` kế thừa `UIScrollView`, hỗ trợ cuộn nội dung khi văn bản dài vượt khung. Thường dùng cho ghi chú, mô tả nhiều dòng. Hỗ trợ tùy chỉnh font, định dạng và cũng có delegate tương tự UITextField [68]. (Có từ iOS 3.0)
- **UIKeyboard** – Lớp đại diện *bàn phím ảo* hệ thống. Mặc dù lập trình viên không trực tiếp khởi tạo `UIKeyboard`, lớp này cung cấp thông tin và notification về bàn phím (như khung bàn phím, sẽ hiển thị/ẩn) để ứng dụng có thể điều chỉnh giao diện khi bàn phím xuất hiện [69]. (Có từ iOS 2.0)
- **UIKeyCommand** – Cho phép định nghĩa *phím tắt bàn phím cứng* (như trên iPad với bàn phím ngoài). Lớp này map một tổ hợp phím (như Cmd+C) thành hành động trong app. Giới thiệu từ iOS 7 để hỗ trợ phím tắt khi có bàn phím vật lý [70].

- **UIPasteboard** – Cung cấp *khả năng copy-paste* với clipboard hệ thống. Thông qua `UIPasteboard.general`, ứng dụng có thể đọc hoặc ghi nội dung (văn bản, hình ảnh, URL...) vào bảng nhớ tạm để chia sẻ giữa các app [71]. (Có từ iOS 3.0)
- **UIPasteConfiguration** – Cung cấp cấu hình về loại dữ liệu mà view chấp nhận dán hoặc kéo thả. Giới thiệu iOS 11 nhằm tăng cường bảo mật pasteboard, cho phép app khai báo rõ loại nội dung mình có thể paste (text, image...) [72]. (Có từ iOS 11.0)
- **UIPasteControl** – *Nút dán nội dung* tiêu chuẩn (giới thiệu iOS 16). Apple cung cấp sẵn control này để người dùng bấm và dán nội dung từ clipboard mà không bị hiện prompt cho phép dán mỗi lần [73]. Control này tự động xử lý xin quyền đọc clipboard trên iOS 16+.
- **UIMenuController** – Quản lý menu *Cut/Copy/Paste* dạng nổi khi người dùng chọn văn bản. Lớp này điều khiển menu chỉnh sửa (với các mục như "Cut, Copy, Paste, Select...") xuất hiện phía trên văn bản được chọn [74]. (Có từ iOS 3.0, **đã deprecated trên iOS 16** – thay bằng `UIEditMenuInteraction`)
- **UIEditMenuInteraction** – Tương tác menu chỉnh sửa *mới từ iOS 16*, thay thế UIMenuController. Cung cấp menu văn bản với giao diện hiện đại và linh hoạt trên iPhone/iPad (bao gồm cả khi dùng chuột phải trên iPad). Để sử dụng, developer đính `UIEditMenuInteraction` vào view văn bản để tự động có menu cut/copy/paste. (Có từ iOS 16.0) [75]
- **UIDocument** – Lớp trừu tượng đại diện cho *tài liệu (document)*, hỗ trợ quản lý dữ liệu tài liệu ở local hoặc iCloud. `UIDocument` tự động quản lý việc đọc/ghi file và tích hợp với cơ chế *Auto Save*, *Versions* của iOS [76]. (Có từ iOS 5.0)
- **UIManagedDocument** – Một subclass của UIDocument kết hợp với Core Data, giúp lưu dữ liệu Core Data ra file tài liệu một cách thuận tiện [77]. (Có từ iOS 5.0)
- **UIDictationPhrase** – Đại diện cho một *cụm từ nhận dạng* được từ tính năng dictation (nhận diện giọng nói). Nếu người dùng dùng chức năng đọc giọng nói để nhập text, hệ thống sẽ trả về một hoặc nhiều `UIDictationPhrase` chứa text đã nhận dạng [78]. (Có từ iOS 5.0)
- **UITextChecker** – *Bộ kiểm tra chính tả*, cho phép kiểm tra lỗi chính tả và gợi ý từ. Lớp này có thể dùng để xây dựng chức năng kiểm tra lỗi chính tả trong ứng dụng (nó dùng từ điển hệ thống để so sánh) [79]. (Có từ iOS 3.0)
- **UITextInputMode** – Đại diện cho *phương thức nhập hiện tại*, chẳng hạn ngôn ngữ bàn phím người dùng đang chọn. Developer có thể lấy `[UITextInputMode currentInputMode]` để biết người dùng đang dùng bàn phím ngôn ngữ gì [80]. (Có từ iOS 4.2)
- **UITextInputAssistantItem** – Quản lý *thanh phím tắt* phía trên bàn phím (shortcut bar, chứa các nút Undo/Redo hoặc nút tùy chỉnh). Cho phép tùy chỉnh các nút *BarButtonItem* xuất hiện trên thanh hỗ trợ bàn phím của `UITextField` hoặc `UITextView` [81]. (Có từ iOS 9.0)
- **UITextPosition** – Đại diện cho *một vị trí* trong nội dung văn bản (ví dụ sau ký tự thứ n). Được dùng trong nội bộ các thành phần nhập liệu để đánh dấu vị trí con trỏ hoặc điểm bắt đầu/kết thúc selection [82].
- **UITextRange** – Đại diện cho *một đoạn range* (khoảng) trong văn bản, bao gồm vị trí bắt đầu và kết thúc (`UITextPosition`). Thường dùng để biểu diễn đoạn văn bản được chọn (selection) hoặc vùng sẽ thay thế [83].
- **UITextSelectionRect** – Lớp trừu tượng biểu diễn *hình chữ nhật vùng chọn văn bản* đang hiển thị trên giao diện. Cung cấp thông tin toạ độ vùng chọn, thường dùng cho xử lý tùy chỉnh việc vẽ hoặc tương tác vùng chọn văn bản [83].
- **UIInputView** – View *bàn phím tùy chỉnh*. Đây là lớp cơ sở cho giao diện bàn phím do developer tạo ra. `UIInputView` có thể được gán làm inputView của một UITextField/TextView để thay bàn phím hệ thống bằng giao diện tùy chỉnh (ví dụ bàn phím emoji riêng) [84]. (Có từ iOS 8.0)
- **UIInputViewController** – *View controller cho bàn phím tùy chỉnh*. Developer tạo subclass của lớp này để triển khai một bộ bàn phím hệ thống (extension) riêng. iOS sẽ quản lý vòng đời và hiển thị `UIInputViewController` khi người dùng chọn bàn phím custom [85]. (Có từ iOS 8.0)

Cử chỉ và Tương tác (Gestures & Interactions)

- **UIGestureRecognizer** – Lớp cơ sở cho tất cả *bộ nhận diện cử chỉ*. Nó theo dõi các chuỗi touch trên view và xác định xem có khớp với một cử chỉ (gesture) nhất định hay không (ví dụ tap, pinch). Khi gesture được nhận diện, nó sẽ gửi hành động tương ứng. Nhiều gesture recognizer có sẵn như tap, swipe, pinch,... đều kế thừa lớp này [86] . (Có từ iOS 3.2)
- **UITapGestureRecognizer** – Nhận diện *cử chỉ chạm* (tap) – bao gồm chạm một hoặc nhiều lần, một hoặc nhiều ngón. Ví dụ dùng nhận biết chạm đơn, chạm đúp để phóng to,... [87] . (Có từ iOS 3.2)
- **UISwipeGestureRecognizer** – Nhận diện *cử chỉ vuốt* ngang hoặc dọc với một hướng nhất định. Có thể cấu hình số ngón tay và hướng vuốt (trái, phải, lên, xuống) [88] . (Có từ iOS 3.2)
- **UIPinchGestureRecognizer** – Nhận diện *cử chỉ "nhúm" 2 ngón* (pinch) thường dùng để phóng to/thu nhỏ nội dung. Cung cấp scale factor để biết mức độ phóng to/thu nhỏ [89] . (Có từ iOS 3.2)
- **UIRotationGestureRecognizer** – Nhận diện *cử chỉ xoay 2 ngón* trên view. Cung cấp góc xoay (radians) khi người dùng dùng hai ngón tay xoay đối tượng trên màn hình [90] . (Có từ iOS 3.2)
- **UIPanGestureRecognizer** – Nhận diện *cử chỉ kéo rê* (pan or drag) – tức là giữ và di chuyển ngón tay trên màn hình. Thường dùng để kéo đối tượng hoặc cuộn tùy chỉnh [91] . (Có từ iOS 3.2)
- **UIScreenEdgePanGestureRecognizer** – Nhận diện *cử chỉ vuốt từ mép màn hình*. Giống UIPan nhưng chỉ khởi phát khi người dùng vuốt bắt đầu từ cạnh màn hình (ví dụ vuốt từ mép trái để quay lại) [92] . (Có từ iOS 7.0)
- **UILongPressGestureRecognizer** – Nhận diện *cử chỉ nhấn giữ* (chạm và giữ lâu tại một điểm). Có thể cấu hình thời gian tối thiểu để tính là nhấn giữ. Thường để kích hoạt menu ngữ cảnh, kéo đối tượng,... [93] . (Có từ iOS 3.2)
- **UIHoverGestureRecognizer** – Nhận diện *cử chỉ hover chuột* (di chuột mà không nhấn) trên iPad (giới thiệu trong iPadOS 13.4 cho trackpad/mouse). Cho phép view phản ứng khi con trỏ di chuyển vào/ra và di chuyển qua lại bên trong vùng của view [94] . (Có từ iOS 13.4 trên iPad)
- **UIDragInteraction** – Cung cấp khả năng *kéo (drag) nội dung* trong giao diện. Khi thêm `UIDragInteraction` vào một view, người dùng có thể nhấn giữ và kéo đối tượng từ view đó để bắt đầu phiên kéo thả. Lớp này quản lý việc khởi tạo drag và liên kết với hệ thống Drag & Drop của iOS [95] . (Có từ iOS 11.0)
- **UIDropInteraction** – Cung cấp khả năng *thả (drop) nội dung* vào view. Khi thêm `UIDropInteraction` vào view đích, view có thể nhận dữ liệu do người dùng kéo thả vào (hình ảnh, văn bản,...). Lớp này quản lý các sự kiện khi nội dung đang được kéo trên view và khi thả nội dung xuống [96] . (Có từ iOS 11.0)
- **UIDragItem** – Đại diện cho *một mục dữ liệu đang được kéo*. Mỗi `UIDragInteraction` khi bắt đầu sẽ gói dữ liệu (item provider) vào các `UIDragItem`. Thường chứa thông tin về đối tượng (như UIImage, NSString) và cho phép cung cấp theo yêu cầu trong quá trình drag [97] . (Có từ iOS 11.0)
- **UIDragPreview / UIDragPreviewParameters** – `UIDragPreview` đại diện cho *hình ảnh xem trước* của đối tượng khi người dùng đang kéo, cho phép tùy chỉnh giao diện (ví dụ bo góc, làm mờ nền). `UIDragPreviewParameters` chứa tham số cấu hình preview (vùng nội dung hiển thị, đường viền trong suốt, v.v.) [98] [99] . (Có từ iOS 11.0)
- **UIDragPreviewTarget** – Xác định *điểm đích* và *hình dạng* mà item kéo sẽ bay tới khi kết thúc kéo hoặc bị hủy. Lớp này dùng để tạo hiệu ứng animation mượt mà khi người dùng thả hoặc bỏ kéo một item, chỉ định vị trí và view đích để animation hướng tới [100] . (Có từ iOS 11.0)
- **UIDropProposal** – Mô tả *cách thức xử lý* khi nội dung được thả vào view đích. Ví dụ cho biết view sẽ nhận như là copy hay move dữ liệu, hay từ chối không cho thả. Lớp này do `UIDropInteractionDelegate` trả về trong quá trình thả để chỉ dẫn hệ thống [101] . (Có từ iOS 11.0)
- **UIContextMenuInteraction** – Cho phép *menu ngữ cảnh kiểu mới (iOS 13)* khi người dùng nhấn giữ (hoặc 3D Touch) một view. Khi thêm `UIContextMenuInteraction` vào view, người dùng

có thể mở menu chứa các hành động (dạng giống menu chuột phải trên desktop) với preview nội dung. Tính năng này thay thế phần nào cho Peek & Pop trên các thiết bị có 3D Touch [102]. (Có từ iOS 13.0)

- **UIContextMenuConfiguration** – Cấu trúc cấu hình cho menu ngữ cảnh (được cung cấp trong delegate của `UIContextMenuInteraction`). Xác định nội dung preview (nếu có) và menu các hành động (`UIMenu`) sẽ hiển thị. (Có từ iOS 13.0)
- **UIPointerInteraction** – Tương tác hỗ trợ *con trỏ chuột và trackpad* trên iPad (giới thiệu iOS 13.4). Gắn `UIPointerInteraction` vào view cho phép tùy biến hiệu ứng con trỏ khi hover trên view đó (ví dụ highlight, phóng to, đổi hình dạng con trỏ) [103]. (Có từ iPadOS 13.4)
- **UIPreviewInteraction** – Cung cấp tương tác *"Peek" (nhấn nhẹ để xem trước)* trên thiết bị hỗ trợ 3D Touch. Cho phép đăng ký một view để nhận sự kiện lực nhấn và phản hồi với giao diện preview nội dung. Kết hợp với `UIPreviewAction` tạo thành tính năng Peek & Pop (đã xuất hiện trên iPhone có 3D Touch, iOS 9) [104]. (Có từ iOS 10.0)
- **UIPreviewAction / UIPreviewActionGroup** – Các *hành động nhanh trong Peek & Pop*. Khi dùng 3D Touch nhấn mạnh (Pop) trên một đối tượng, các `UIPreviewAction` sẽ hiện ra dưới dạng danh sách cho người dùng chọn (ví dụ "Share", "Delete"...). `UIPreviewActionGroup` cho phép nhóm nhiều hành động lại trong một menu con [105]. (Có từ iOS 9.0)
- **UIFocusGuide** – Hướng dẫn *điều hướng focus* (đặc biệt hữu ích trên tvOS hoặc khi dùng bàn phím điều khiển trên iPad). `UIFocusGuide` là một vùng vô hình có thể lấy focus, nhằm chuyển hướng focus đến một view khác khi người dùng di chuyển (bằng phím/remote) [106]. (Có từ iOS 9.0, iOS 10 hỗ trợ trên iPad)
- **UIFocusSystem** – Quản lý *trạng thái focus hiện tại* trong giao diện (đặc biệt cho tvOS và iPad với keyboard). Cho phép truy xuất view hiện đang được focus, cập nhật focus theo ngữ cảnh... [107]. (Có từ iOS 10.0)
- **UIFocusAnimationCoordinator** – Điều phối *hiệu ứng hoạt họa khi chuyển focus*. Khi focus thay đổi từ view này sang view khác, `UIFocusAnimationCoordinator` cho phép chạy các animation đồng bộ (ví dụ làm mờ, phóng to view được focus) để chuyển tiếp mượt mà [108].
- **UIFocusUpdateContext** – Chứa *thông tin về lần cập nhật focus* (focus chuyển từ view nào sang view nào). Lớp này được truyền vào các phương thức delegate khi focus thay đổi, giúp app biết đối tượng mất focus và được focus [109].

Hoạt ảnh và Động lực học (Animation & Dynamics)

- **UIView (Animation)** – (Không phải một class riêng, mà là **chức năng hoạt ảnh** có sẵn của UIView). UIKit cung cấp các hàm tiện ích trên `UIView` để thực hiện *hoạt ảnh* chuyển đổi thuộc tính giao diện (như alpha, frame) một cách mượt mà trong một khoảng thời gian nhất định. Ví dụ: `UIView.animate(withDuration: animations: ...)`. (Có từ iOS 2.0, mở rộng nhiều từ iOS 4.0 với block-based animation)
- **UIDynamicAnimator** – Bộ *điều phối các hiệu ứng vật lý* (UIKit Dynamics). Bạn tạo một `UIDynamicAnimator` gắn với một UIView cụ thể, sau đó thêm các `UIDynamicBehavior` (hành vi động) vào để áp dụng cho các item (các view) bên trong animator đó. Animator sẽ cập nhật vị trí các view theo mô phỏng vật lý (trọng lực, va chạm, v.v.) [110]. (Có từ iOS 7.0)
- **UIDynamicBehavior** – Lớp cơ sở cho *các hành vi động* mô phỏng vật lý. UIKit cung cấp sẵn nhiều hành vi con như trọng lực, va chạm, gắn lò xo,... Bạn cũng có thể tùy biến subclass. Một `UIDynamicBehavior` có thể gắn vào `UIDynamicAnimator` để ảnh hưởng lên các phần tử động [111]. (Có từ iOS 7.0)
- **UIGravityBehavior** – *Mô phỏng trọng lực* kéo các vật thể (các item động) theo một vector gia tốc cố định (mặc định hướng xuống). Có thể điều chỉnh hướng và độ lớn gia tốc để tạo hiệu ứng rơi [112].

- **UICollisionBehavior** – *Mô phỏng va chạm*. Cho phép các vật thể tương tác va chạm với nhau hoặc với ranh giới (có thể là khung view). Lớp này theo dõi và phát hiện khi các `UIDynamicItem` chạm vào nhau hoặc chạm vào biên đã đặt ra [113].
- **UIAttachmentBehavior** – *Liên kết* một vật thể động với một điểm cố định hoặc với một vật thể động khác. Ví dụ: gắn một view vào một điểm neo bằng lò xo, hoặc nối hai view bằng một thanh nối ảo. Có nhiều loại liên kết (kéo căng, giữ cố định khoảng cách, v.v.) [114].
- **UISnapBehavior** – *Neo vật thể* về một điểm với hiệu ứng *bật lò xo*. Khi khởi tạo với một điểm mục tiêu, `UISnapBehavior` sẽ kéo view về điểm đó với lực giảm dần (giống thả một vật gắn lò xo rồi nó dao động về vị trí cân bằng) [115].
- **UIPushBehavior** – *Đẩy vật thể* với một lực theo hướng xác định. Có thể cấu hình đẩy tức thời (instantaneous) hoặc đẩy liên tục (continuous) để làm vật di chuyển theo vector cho trước, tương tự áp lực lên vật trong thế giới thực [116].
- **UIFieldBehavior** – Mô phỏng *trường lực* (field) tác động lên các vật thể trong vùng, ví dụ trường trọng lực, lực điện, lực xoáy. Bạn có thể tạo trường hấp dẫn, đẩy, nhiễu loạn,... tác động đồng thời lên nhiều vật trong một khu vực thông qua `UIFieldBehavior` [117].
- **UIDynamicItemBehavior** – Cho phép *tùy chỉnh thuộc tính vật lý* của các vật thể động (UIDynamicItem). Ví dụ: đặt trọng lượng, độ đàn hồi (bật nảy), ma sát, khả năng chịu quay... Lớp này cũng cho phép thêm *vận tốc* tùy ý cho vật (ném vật đi với vận tốc đầu) [118].
- **UIRegion** – Xác định *vùng hình học* (hình tròn, hình chữ nhật hoặc do vẽ Path) để làm vùng ảnh hưởng của `UIFieldBehavior` hoặc các tính năng hit-testing khác. (Có từ iOS 9.0) [119]
- **UIViewPropertyAnimator** – Lớp mới (iOS 10) để điều khiển *hoạt ảnh các thuộc tính UIView* một cách linh hoạt. Cho phép *tạm dừng, tiếp tục, đảo chiều* animation, cũng như liên kết animation với các tương tác người dùng (ví dụ cuộn hay kéo thả). `UIViewPropertyAnimator` quản lý một hoặc nhiều animation property của view và cho phép thay đổi tiến trình animation trong runtime [120]. (Có từ iOS 10.0)
- **UISpringTimingParameters** – Cung cấp *thông số animation kiểu lò xo*, dùng để tạo animator có hiệu ứng đàn hồi. Bạn có thể cấu hình độ mạnh lò xo, độ cản (damping), v.v. rồi đưa vào `UIViewPropertyAnimator` để có animation mềm mại, tự nhiên hơn [121]. (Có từ iOS 10.0)
- **UICubicTimingParameters** – Cung cấp *thông số animation dưới dạng đường cong Bézier*. Cho phép định nghĩa đường cong tăng tốc/giảm tốc tùy chỉnh cho hoạt ảnh UIView (tương tự các curve EaseIn, EaseOut...). (Có từ iOS 10.0) [122]
- **UIPercentDrivenInteractiveTransition** – Lớp hỗ trợ *animation chuyển cảnh view controller tương tác*. Thường dùng trong các transition tùy chỉnh giữa view controllers, cho phép điều khiển tiến độ chuyển đổi (thường bằng cử chỉ người dùng, ví dụ vuốt để dismiss view). Developer cập nhật thuộc tính percentComplete để điều khiển hoạt ảnh theo input thời gian thực [123]. (Có từ iOS 7.0)
- **UIFeedbackGenerator** – Lớp cơ sở cho *phản hồi xúc giác (haptic feedback)*. Bắt đầu từ iOS 10, iPhone có Taptic Engine cho phép tạo rung phản hồi tinh tế. Các subclass như `UIImpactFeedbackGenerator`, `UINotificationFeedbackGenerator`, `UISelectionFeedbackGenerator` tạo ra các kiểu rung nhẹ khi chạm hoặc sự kiện xảy ra [124]. (Có từ iOS 10.0)
- **UIImpactFeedbackGenerator** – Tạo *phản hồi rung dạng va chạm*. Ví dụ rung nhẹ khi một đối tượng "chạm" vào đối tượng khác. Bạn có thể chọn độ nặng (light, medium, heavy) để mô phỏng cảm giác va chạm mạnh nhẹ khác nhau [125].
- **UINotificationFeedbackGenerator** – Tạo *phản hồi rung cho thông báo*, tương ứng các kiểu thành công, thất bại, cảnh báo. Mỗi kiểu có pattern rung khác nhau (ví dụ thành công rung nhẹ hai nhịp) [126].
- **UISelectionFeedbackGenerator** – Tạo *phản hồi rung cho lựa chọn thay đổi*. Mỗi khi người dùng thay đổi giá trị lựa chọn (như cuộn picker), generator này phát một rung *tick* nhẹ để xác nhận thay đổi [127].

Đồ họa và Vẽ (Graphics & Drawing)

- **UIColor** – Đại diện cho *màu sắc* (và họa tiết) trong UIKit. Cung cấp các màu cơ bản, màu hệ thống, và có thể tạo màu tùy ý (RGB, HSB, pattern image...). `UIColor` được dùng để đặt màu nền, màu chữ, màu nét vẽ,... cho các thành phần giao diện [128]. (Có từ iOS 2.0)
- **UIImage** – Đại diện cho *hình ảnh* (bitmap hoặc vector) trong ứng dụng. `UIImage` có thể được tạo từ file (PNG, JPEG...), từ dữ liệu, hoặc từ vẽ CoreGraphics. Nó hỗ trợ các thao tác cơ bản như resize, dựng ảnh CIImage, v.v. và là đối tượng được `UIImageView` hiển thị [129]. (Có từ iOS 2.0)
- **UIBezierPath** – Lớp hỗ trợ tạo *đường path vector* (các đoạn thẳng, cong, hình học) để vẽ đồ họa 2D. Thay vì dùng trực tiếp CGPath, `UIBezierPath` cung cấp API hướng đối tượng để vẽ các hình như đường cong Bézier, hình tròn, chữ nhật,... Dùng trong vẽ Core Graphics hoặc để tạo vùng hit-test tùy ý [130]. (Có từ iOS 3.2)
- **UIGraphicsImageRenderer** – Lớp tiện ích (iOS 10) để *vẽ vào một đối tượng UIImage* dễ dàng. Bạn khởi tạo renderer với kích thước và tùy chọn, sau đó dùng phương thức `renderer.image { context in ... }` với các lệnh vẽ Core Graphics bên trong, renderer sẽ tạo ra UIImage kết quả [131].
- **UIGraphicsImageRendererFormat** – Định nghĩa *định dạng đầu ra* cho UIGraphicsImageRenderer (ví dụ scale @2x, chế độ màu, opaque hay không).
- **UIGraphicsImageRendererContext** – Ngữ cảnh vẽ cung cấp các phương thức vẽ CoreGraphics trong block renderer (được truyền vào closure). Cho phép truy cập `cgContext` và thêm tiện ích chuyển đổi sang UIImage khi vẽ xong [132].
- **UIGraphicsPDFRenderer** – Tương tự ImageRenderer nhưng dùng để *vẽ nội dung PDF*. Bạn có thể tạo file PDF bằng cách vẽ trang trong closure `renderer.pdfData { context in ... }`. Lớp này tự quản lý việc tạo nhiều trang PDF và dữ liệu PDF đầu ra [133].
- **UIGraphicsPDFRendererFormat** – Định dạng đầu ra cho PDF (ví dụ thông tin metadata PDF, kích thước trang).
- **UIGraphicsPDFRendererContext** – Ngữ cảnh vẽ trang PDF, cung cấp phương thức bắt đầu trang mới, ghi metadata, v.v. trong quá trình vẽ [134].
- **UIGraphicsRenderer** – Lớp cơ sở chung cho ImageRenderer và PDFRenderer (không dùng trực tiếp). Định nghĩa các thuộc tính cơ bản của một đối tượng renderer và quản lý ngữ cảnh đồ họa [135].
- **UIPrintFormatter** – Lớp trừu tượng để *dàn trang in ấn*. Các nội dung cần in (text, hình ảnh) có thể được đóng gói vào subclass của `UIPrintFormatter` để UIPrintPageRenderer biết cách vẽ chúng lên trang giấy [136].
- **UISimpleTextPrintFormatter** – Format đơn giản để in chuỗi văn bản thuần. Cho phép định dạng font, màu chữ, align... một cách đồng nhất cho toàn bộ văn bản [137]. (Có từ iOS 4.2)
- **UIMarkupTextPrintFormatter** – Format để in nội dung HTML hoặc NSAttributedString. Lớp này sẽ lo việc bố trí và vẽ HTML/text dạng markup thành nhiều trang nếu cần thiết [138]. (Có từ iOS 4.2)
- **UIPrintPageRenderer** – Lớp chịu trách nhiệm *vẽ nội dung từng trang in*. Bạn có thể subclass lớp này để tùy biến cách vẽ header, footer, nội dung trang. Mặc định, `UIPrintPageRenderer` kết hợp với các `UIPrintFormatter` để phân chia nội dung vào các trang giấy [139]. (Có từ iOS 4.2)
- **UIPrintPaper** – Đại diện cho *kích cỡ giấy in* và vùng in được trên trang giấy đó. Hệ thống cung cấp các kích cỡ chuẩn (A4, Letter,...). `UIPrintPaper` có thuộc tính kích thước tổng và vùng margin khả dụng [140].
- **UIPrintInfo** – Chứa *thông tin cấu hình cho job in ấn*, ví dụ loại đầu ra (đen trắng hay màu), số bản sao, hướng giấy,... App có thể thiết lập `UIPrintInfo` rồi gắn vào `UIPrintInteractionController` trước khi in [141]. (Có từ iOS 4.2)
- **UIPrintInteractionController** – *Trình điều khiển in ấn chính* trên iOS. Lớp này quản lý việc tương tác với hệ thống in: chọn máy in (trên Wi-Fi), gửi dữ liệu tới máy in, hiển thị giao diện chọn máy

in. Developer thiết lập các `UIPrintPageRenderer` hoặc trực tiếp `printingItem` (dữ liệu cần in) cho nó rồi gọi present để hiển thị UI in [142]. (Có từ iOS 4.2)
- **UIPrinter** – Thông tin về *máy in AirPrint* (tên, trạng thái, chế độ). Lớp này cho phép liệt kê và chọn máy in lập trình, hoặc kiểm tra các thuộc tính máy in đã chọn [143]. (Có từ iOS 8.0)
- **UIPrinterPickerController** – Giao diện UI cho phép người dùng *chọn máy in* có sẵn trong mạng. Khi present controller này, nó sẽ quét các máy in AirPrint và cho phép chọn một máy in. Kết quả lựa chọn được trả về qua completion handler [144]. (Có từ iOS 8.0)
- **UIRegion** – *(Đã liệt kê ở phần Dynamics ở trên)* Vùng hình học dùng để xác định phạm vi hiệu lực cho một số thao tác, ví dụ giới hạn vùng ảnh hưởng của UIFieldBehavior.
- **UIMenu** – Lớp đại diện cho *menu gồm nhiều hành động* (giới thiệu iOS 13). Một `UIMenu` có thể chứa các `UIAction` hoặc các menu con, sử dụng trong context menu hoặc trên thanh menu của iPad.
- **UIAction** – Lớp giới thiệu iOS 13 đi kèm UIMenu, đại diện cho *một hành động* có tiêu đề, image, và handler code. Khác với UIBarButtonItem, `UIAction` chủ yếu dùng để cấu hình các mục trong UIMenu, UIContextMenu.

(Lưu ý: Các lớp NS như NSLayoutConstraint, NSStringDrawing, Core Animation... không thuộc UIKit nên không liệt kê. Ngoài ra, các protocol (I/UINavigationControllerDelegate...) cũng không được liệt kê vì không phải class.)

Hỗ trợ & Khác (Accessibility & Others)

- **UIAccessibility** – Cung cấp các *API hỗ trợ tiếp cận* (Accessibility) cho UIKit. Thực ra đây là **họ** phương thức mở rộng (category) trên `NSObject` giúp gắn nhãn, hint, v.v... chứ không phải class cụ thể. Nhờ UIAccessibility, các đối tượng UIView có thể phục vụ VoiceOver và các công nghệ hỗ trợ [145].
- **UIAccessibilityElement** – Đại diện cho *một phần tử giao diện dành cho Accessibility*. Dùng để mô tả những thành phần không phải UIView nhưng cần VoiceOver đọc được (ví dụ một vùng custom vẽ lên). Bạn tạo `UIAccessibilityElement`, gán nhãn, khung, hint,... để VoiceOver coi nó như một đối tượng giao diện thật [146]. (Có từ iOS 3.0)
- **UIAccessibilityCustomAction** – Cho phép thêm *hành động tùy chỉnh* vào menu VoiceOver cho một phần tử. Ví dụ, một view có thể có hành động "Yêu thích" ngoài các hành động mặc định. Developer tạo `UIAccessibilityCustomAction` với tên và selector, rồi gắn vào phần tử tương ứng [147]. (Có từ iOS 9.0)
- **UIAccessibilityCustomRotor** – Được giới thiệu iOS 10, cho phép tạo *bộ "Rotor" tùy chỉnh* trong VoiceOver. Rotor là menu xoay cho phép người dùng duyệt nhanh qua nội dung (ví dụ tiêu đề, liên kết). Với lớp này, app có thể tạo rotor riêng (ví dụ rotor để duyệt qua các comment trong màn hình) [148].
- **UIAccessibilityCustomRotorItemResult** và **UIAccessibilityCustomRotorSearchPredicate** – Đi kèm CustomRotor, lớp **ItemResult** đại diện cho mục tiêu tiếp theo của rotor (ví dụ phần tử kế tiếp phù hợp với tiêu chí), còn **SearchPredicate** mô tả tiêu chí tìm kiếm (ví dụ tìm chuỗi kế tiếp thỏa mãn) [149].
- **UIAccessibilityContainerDataTable** – Lớp ít gặp, giới thiệu iOS 11 để hỗ trợ VoiceOver mô tả *bố cục bảng (table)* phức tạp. Developer có thể dùng lớp này để cung cấp thông tin về số hàng, số cột, tiêu đề hàng/cột,... của một bảng vẽ custom, giúp VoiceOver thông báo đúng nội dung ô đang đọc [150].
- **UIAccelerometer** – *Cảm biến gia tốc* (legacy). Cho phép nhận dữ liệu gia tốc (x, y, z) từ phần cứng thiết bị. Lớp này có singleton `UIAccelerometer.shared` và bạn có thể gán delegate để nhận cập nhật gia tốc liên tục [151]. **Đã bị deprecate từ iOS 5.0** – Apple khuyến cáo chuyển sang **CoreMotion** (`CMMotionManager`) để lấy gia tốc và con quay hồi chuyển [152]. (Có từ iOS 2.0, ngừng iOS 5.0)

- **UIAcceleration** – Cấu trúc (struct) chứa dữ liệu *gia tốc tức thời* trên mỗi trục, được `UIAccelerometer` sử dụng khi gọi delegate. (Có từ iOS 2.0)
- **UIUserNotificationSettings** – Cấu hình loại thông báo (alert, âm thanh, badge) mà app đăng ký với hệ thống (iOS 8 & 9). Developer dùng lớp này để yêu cầu quyền hiển thị thông báo. **Đã deprecated iOS 10.0** khi framework UserNotifications ra đời [153]. (Có từ iOS 8.0, ngừng iOS 10.0)
- **UIUserNotificationAction** và **UIUserNotificationCategory** – Cũng giới thiệu iOS 8 để hỗ trợ *tương tác trên thông báo* (như nút "Reply" trên banner). `UIUserNotificationAction` định nghĩa nút hành động, và `UIUserNotificationCategory` gom nhóm các hành động cho một loại thông báo. **Đã thay thế bằng UNNotificationAction/Category từ iOS 10**.
- **UILocalNotification** – Lớp đại diện cho *thông báo cục bộ* lên thiết bị (không qua server). App có thể tạo `UILocalNotification`, đặt thời gian hoặc lặp lại, nội dung, âm thanh... để hệ thống hiển thị cho người dùng vào thời điểm đặt trước. **Đã deprecated iOS 10** (thay bằng UNNotificationRequest) [154]. (Có từ iOS 4.0)
- **UIApplicationShortcutItem** – Đại diện cho *Quick Action* (3D Touch) trên icon ứng dụng (giới thiệu iOS 9). App có thể định nghĩa các shortcut tĩnh trong Info.plist hoặc tạo động bằng lớp này. Mỗi `UIApplicationShortcutItem` gồm type, tiêu đề, icon,... và khi người dùng nhấn mạnh vào icon app sẽ hiện menu các shortcut, chọn mục nào thì app nhận được qua delegate [155]. (Có từ iOS 9.0)

Tài liệu tham khảo: - Apple Developer Documentation – *UIKit Framework Reference* [39] [156]
- Xamarin iOS (Microsoft Learn) – *UIKit Namespace Reference* (mô tả các lớp UIKit) [125] [90]
- NSHipster – *UIStackView Essentials* (giới thiệu UIStackView iOS 9) [35]
- Five Stars Blog – *Context Menus & UIContextMenuInteraction* [102]
- Stack Overflow – *UIAccelerometer deprecated since iOS 5* [152], *UIWebView deprecated iOS 12* [23], *UIUserNotificationSettings deprecated iOS 10* [153].

---

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 24 25 26 27 28 29 30 31 32 33 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 74 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 95 96 97 98 99 100 101 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 154 155 156 UIKit Namespace | Microsoft Learn
https://learn.microsoft.com/en-us/dotnet/api/uikit?view=xamarin-ios-sdk-12

23 Why am I seeing a "Couldn't Connect to the Duo Mobile App" error ...
https://help.duo.com/s/article/8257

34 35 UIStackView - NSHipster
https://nshipster.com/uistackview/

73 Everything You Need to Know About iOS 16 and Pasteboard Opt-Ins
https://www.branch.io/resources/blog/everything-you-need-to-know-about-ios-16-and-pasteboard-opt-ins/

75 How to use UIEditMenu with WKWebView in iOS 16? - Stack Overflow
https://stackoverflow.com/questions/74752168/how-to-use-uieditmenu-with-wkwebview-in-ios-16

94 Catalyst by Tutorials, Chapter 9: The Mouse - Kodeco
https://www.kodeco.com/books/catalyst-by-tutorials/v1.0/chapters/9-the-mouse

102 Behind The Scenes Of Context Menus | FIVE STARS
https://www.fivestars.blog/articles/uicontextmenuinteraction/

103 UIPointerInteraction | Apple Developer Documentation
https://developer.apple.com/documentation/uikit/uipointerinteraction

152 How to replace UIAccelerometer with CMMotionManager?
https://stackoverflow.com/questions/20340384/how-to-replace-uiaccelerometer-with-cmmotionmanager

153 UIUserNotificationSettings | Apple Developer Documentation
https://developer.apple.com/documentation/uikit/uiusernotificationsettings