Shape, square

Description automatically generated

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

VIỆT - HÀN

**Khoa Khoa Học Máy Tính**



ĐỒ ÁN CƠ SỞ 5

**ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG CHỈNH SỬA**

**ẢNH ĐA NĂNG TRÊN IOS**

Sinh viên thực hiện: **NGUYỄN ĐĂNG QUÝ**

Lớp: **19IT4**

Giảng viên hướng dẫn: **THS. TRỊNH THỊ NGỌC LINH**

***Đà nẵng, tháng 05 năm 2022***

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

VIỆT - HÀN

**Khoa Khoa Học Máy Tính**



ĐỒ ÁN CƠ SỞ 5

**ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG CHỈNH SỬA ẢNH ĐA NĂNG TRÊN IOS**

Sinh viên thực hiện: **NGUYỄN ĐĂNG QUÝ**

Lớp: **19IT4**

Giảng viên hướng dẫn: **THS. TRỊNH THỊ NGỌC LINH**

**Đà Nẵng, tháng 05 năm 2022**

**LỜI CẢM ƠN**

Đồ án này được được thực hiện bởi nhóm gồm 1 thành viên thuộc sinh viên của trường Đại học Công nghệ thông tin và Truyền thông Việt – Hàn Đà Nẵng. Trước hết, xin gởi lời cảm ơn chân thành đến **Cô giáo Trịnh Thị Ngọc Linh** đã tận tình chỉ dẫn, góp ý để báo cáo của nhóm tránh nhiều sai sót và hoàn thiện hơn. Tiếp đến, xin gởi lời cảm ơn đến các tác giả bài viết, các trang web đã góp phần cung cấp cho chúng tôi các tiện ích và thông tin cần thiết để thực hiện đồ án.

Cuối cùng, mặc dù nhóm đã rất cố gắng nhưng chắc chắn sẽ còn rất nhiều sai sót mà chưa thể khắc phục hết được vì vậy rất mong Cô giáo đóng góp ý kiến, phê bình để nhóm rút thêm kinh nghiệm cho những lần sau.

**Xin chân thành cảm ơn!**

**NHẬN XÉT**

**(Của giảng viên hướng dẫn)**

…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

Đà Nẵng, ngày … tháng ... năm 2022

**Giảng viên hướng dẫn**

**ThS. Trịnh Thị Ngọc Linh**

**MỤC LỤC**

Trang

[Chương 1 TỔNG QUAN 1](#_Toc105501061)

[1.1 Giới thiệu 1](#_Toc105501062)

[1.2 Lý do chọn đề tài 1](#_Toc105501063)

[1.3 Mục tiêu 2](#_Toc105501064)

[1.4 Cấu trúc đồ án 2](#_Toc105501065)

[Chương 2 CƠ SỞ LÝ THUYẾT 4](#_Toc105501066)

[2.1 Tổng quan về ngôn ngữ Swift 4](#_Toc105501067)

[2.1.1 Các đặc điểm của ngôn ngữ Swift 4](#_Toc105501068)

[2.1.2 Tìm hiểu về SwiftUI 5](#_Toc105501069)

[2.2 Thư viện hỗ trợ dự án 6](#_Toc105501070)

[2.2.1 SwiftUI: 6](#_Toc105501071)

[2.2.2 Core Image: 6](#_Toc105501072)

[2.2.3 PencilKit 8](#_Toc105501073)

[2.3 Kiến trúc tổng quan 8](#_Toc105501074)

[2.3.1 Giới thiệu mô hình MVVM 8](#_Toc105501075)

[2.3.2 Ưu nhược điểm của MVVM 9](#_Toc105501076)

[Chương 3 PHÂN TÍCH THIẾT KẾ HỆ THỐNG 10](#_Toc105501077)

[3.1 Giới thiệu Xcode 10](#_Toc105501078)

[3.2 Xác định các tác nhân 11](#_Toc105501079)

[3.2.1 Sơ đồ ca sử dụng hệ thống 11](#_Toc105501080)

[3.2.2 Sơ đồ ca sử dụng cho người dùng 12](#_Toc105501081)

[3.2.3 Mô tả ca sử dụng 13](#_Toc105501082)

[3.3 Triển khai xây dựng 14](#_Toc105501083)

[3.3.1 Nhập thư viện 14](#_Toc105501084)

[3.3.2 Cây thư mục Project 15](#_Toc105501085)

[3.3.3 Code giao diện chính 16](#_Toc105501086)

[3.4 Phân tích thuật toán 16](#_Toc105501087)

[3.4.1 Áp dụng bộ lọc 16](#_Toc105501088)

[3.4.2 Thay đổi giá trị bộ lọc 18](#_Toc105501089)

[3.4.3 Vẽ vào ảnh 19](#_Toc105501090)

[3.5 Kết quả cài đặt 22](#_Toc105501091)

[3.5.1 Giao diện chính ứng dụng 22](#_Toc105501092)

[3.5.2 Giao diện chọn ảnh từ thư viện 23](#_Toc105501093)

[3.5.3 Giao diện chỉnh sửa ảnh 23](#_Toc105501094)

[3.5.4 Giao diện vẽ vào ảnh 24](#_Toc105501095)

**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN………………………………………….25**

1. KẾT QUẢ ĐẠT ĐƯỢC………………………………………………25
2. HẠN CHẾ…………………………………………………………….26
3. HƯỚNG PHÁT TRIỂN………………………………………………26

**DANH MỤC TÀI LIỆU THAM KHẢO……………………………………………27**

**DANH MỤC CÁC BẢNG**

[Bảng 3.1 - Chức năng chỉnh ảnh theo bộ lọc 13](#_Toc105501098)

[Bảng 3.2 - Chức năng chỉnh ảnh tuỳ chỉnh 13](#_Toc105501099)

[Bảng 3.3 - Chức năng vẽ vào ảnh 14](#_Toc105501100)

**DANH MỤC CÁC HÌNH**

[Hình 1.1 - Mẫu ứng dụng chỉnh sửa ảnh hiện nay 1](#_Toc105501103)

[Hình 2.1 - Quá trình Core Image xử lý ảnh 7](#_Toc105501104)

[Hình 2.2 - Các tầng mô hình MVVM 9](#_Toc105501105)

[Hình 3.1 - Logo phần mềm Xcode 10](#_Toc105501106)

[Hình 3.2 - Sơ đồ ca sử dụng hệ thống 11](#_Toc105501107)

[Hình 3.3 - Sơ đồ ca sử dụng cho người dùng 12](#_Toc105501108)

[Hình 3.4 - Cây thư mục Project 15](#_Toc105501109)

[Hình 3.5 - Code giao diện chính 16](#_Toc105501110)

[Hình 3.6 - Kết quả sau khi áp dụng CIImage và CIFilter 17](#_Toc105501111)

[Hình 3.7 - Các chức năng cung cấp bởi PKToolPicker 19](#_Toc105501112)

[Hình 3.8 - Giao diện chính ứng dụng 22](#_Toc105501113)

[Hình 3.9 - Giao diện chọn ảnh từ thư viện 23](#_Toc105501114)

[Hình 3.10 - Giao diện chỉnh sửa ảnh 23](#_Toc105501115)

**DANH MỤC CỤM TỪ VIẾT TẮT**

|  |  |
| --- | --- |
| **VIẾT TẮT** | **NỘI DUNG** |
| CNTT | Công nghệ thông tin |
| Obj-C | Objective-C |
| AR | Augmented Reality |
| MVVM | Model - View - ViewModel |

# TỔNG QUAN

## Giới thiệu

* Các ứng dụng truyền thông xã hội ngày càng trở nên rộng rãi hơn. Các ứng dụng chỉnh sửa ảnh cũng trở nên phổ biến trên toàn thế giới vì người dùng muốn đăng những hình ảnh tuyệt vời và sáng tạo của riêng bản thân. Tuy nhiên, người dùng có xu hướng có một loại rào cản tinh thần đối với các ứng dụng trả phí nếu muốn sử dụng các chức năng nâng cao hoặc xoá bỏ logo bản quyền thương hiệu.
* Người dùng thường cố gắng tìm các ứng dụng miễn phí có chức năng tương tự. Vì vậy, chúng ta cần một ứng dụng phải tích hợp một số tính năng độc đáo và thú vị để thuyết phục người dùng. Ngoài ra, nhà phát triển ứng dụng cần cung cấp những tính năng bộ lọc nâng cao hoàn toàn miễn phí.



Hình 1.1 - Mẫu ứng dụng chỉnh sửa ảnh hiện nay

## Lý do chọn đề tài

* Có rất nhiều lý do để chỉnh sửa ảnh của bạn. Đối với hầu hết mọi người, lý do lớn nhất là sửa chữa một sai lầm. Giả sử bạn đã chụp được một bức ảnh tuyệt vời nhưng có một số vết bẩn trên ống kính của bạn. Bây giờ bạn có một điểm tối trên ảnh của bạn. Với ứng dụng chỉnh sửa của mình, bạn có thể quay lại và sử dụng công cụ được gọi là công cụ “xoá mờ” để che đi vết đen đó. Ngoài ra, ảnh của bạn có vẻ hơi tối. Bạn có thể dễ dàng làm sáng ảnh bằng cách thay đổi mức “độ sáng”. Bây giờ bức ảnh của bạn là hoàn hảo! Một lý do khác để bạn có thể chỉnh sửa ảnh của mình là tạo ra các hiệu ứng nghệ thuật. Ví dụ: có thể hình ảnh của bạn có vẻ hơi “bận rộn”. Điều này có thể có nghĩa là sử dụng hiệu ứng "làm mờ" để tạo điểm lấy nét thu hút mắt người nhìn. Ví dụ: nếu bạn có một bức ảnh chụp một đám lá rơi trên cỏ, nhưng tất cả chúng đều được lấy nét và nó có vẻ như mất tập trung, bạn có thể sử dụng tính năng làm mờ để làm mờ một số chiếc lá, về cơ bản là thay đổi "độ sâu trường ảnh" của ảnh của bạn. Bạn đang tạo độ sâu trường ảnh ngắn hơn và chỉ tập trung vào một vùng của hình ảnh, điều này tạo ra thứ gì đó có cảm giác hoàn toàn khác.
* Với phần mềm chỉnh sửa ảnh của mình, bạn có thể tiết lộ tiềm năng thực sự của các bức ảnh của mình. Sử dụng các kỹ năng bạn sẽ học được trong dự án này, bạn sẽ có thể điều khiển ảnh của mình theo bất kỳ cách nào bạn muốn, làm cho màu sắc sống động hơn, làm cho tiêu điểm trở nên sắc nét hơn nhiều và thậm chí mang ảnh trở lại từ bờ vực của thư mục thùng rác. Tất cả những gì bạn cần là một chiếc điện thoại và ứng dụng chỉnh sửa ảnh.

## Mục tiêu

Hỗ trợ chỉnh sửa ảnh cơ bản như:

* Tải ảnh lên, sử dụng bộ lọc
* Chỉnh sửa tuỳ chỉnh
* Vẽ vào ảnh
* Lưu và chia sẻ

## Cấu trúc đồ án

* Dưới đây báo cáo được trình bày trong 4 chương, cụ thể như sau:

Chương 1: Tổng quan. Trong chương này, báo cáo trình bày các mục tiêu, lý do chọn đề tài, mục đích nghiên cứu, ý nghĩa khoa học

Chương 2: Cơ sở lý thuyết. Trong chương này, báo cáo trình bày các khái niệm, phương pháp kết quả

Chương 3: Phân tích và xây dựng hệ thống. Bao gồm các tài liệu liên quan đến quá trình phân tích hệ thống như: phân tích các yêu cầu chức năng và phi chức năng

Chương 4: Triển khai thực hiện. Kiểm thử ứng dụng trên thực tế

Chương 5: Kết luận và hướng phát triển. Đưa ra kết luận và đề xuất các hướng mở rộng, phát triển trong tương lai

Cuối cùng là Danh mục tài liệu tham khảo liên quan đến đề tài.

# CƠ SỞ LÝ THUYẾT

## Tổng quan về ngôn ngữ Swift

Ra đời cách đây 8 năm (từ năm 2014), Swift vẫn là một ngôn ngữ khá trẻ và mới mẻ. Rất nhiều lập trình viên vẫn còn thắc mắc ngôn ngữ lập trình Swift là gì? Tuy nhiên, sự thắc mắc đó dần dần được thay thế theo thời gian, số lượng lập trình viên chọn lựa Swift ngày càng tăng lên, cộng đồng phát triển và hỗ trợ cho ngôn ngữ này ngày càng lớn mạnh.

### Các đặc điểm của ngôn ngữ Swift

* **Khả năng phát triển ứng dụng di động nhanh và dễ dàng: Các lập trình viên không phải mất công chạy các chương trình compiler và các chương trình test một cách thường xuyên nữa, mà thay vào đó tập trung vào các phần tích hợp với mức độ khó hơn. Điều này cũng giúp tiết kiệm sức mạnh và sự hao mòn của phần cứng cho các nhà phát triển.**
* **Code của ngôn ngữ Swift ngắn, gọn và dễ đọc.** Cụ thể, 3 – 5 dòng code của ngôn ngữ Obj-C có thể được viết chỉ trong 1 dòng code của Swift. Các lớp đối tượng trong Swift được đơn giản hóa, các dòng mã được sắp xếp một cách hợp lý và logic hơn. Điều này giúp cho các lập trình viên tiết kiệm rất nhiều thời gian hoàn thành dự án, cũng như tăng hiệu quá trong quá trình bảo trì và fix các lỗi bug trong tương lai. Swift kế thừa và phát huy tất cả các sức mạnh của Objective-C để lại, nên các nhà lập trình sẽ có một môi trường runtime quen thuộc tuyệt vời, dễ dàng quản lý và phát triển.
* Các ứng dụng được viết ra bởi Swift tương thích và ít gây hao mòn phần cứng hơn Obj-C, điều này giúp người dùng có được những trải nghiệm tuyệt vời hơn khi chơi game trên các thiết bị iOS. Bên cạnh những ưu điểm, thì Swift cũng tồn tại một số điểm yếu như sau:
* Do tuổi đời còn quá trẻ, nên việc bắt đầu học Swift cũng khá khó khăn, các nhà lập trình sẽ phải học lại một ngôn ngữ hoàn toàn mới, thậm chí khó cho những người mới bắt đầu hoặc chưa từng học qua Obj-C với nền tảng căn bản.
* Cộng đồng hỗ trợ và phát triển ứng dụng chưa nhiều: có lẽ Apple sẽ mất một khoản thời gian kha khá để xây dựng một cộng đồng sử dụng Swift hùng hậu.
* Do việc phát triển ứng dụng khá nhanh do được tích hợp nhiều frameworks mạnh mẽ, điều này sẽ dẫn tới nhiều ứng dụng chất lượng kém hoặc rác do những lập trình viên mới vào nghề tạo ra Mặc dù còn tồn tại những mặt khuyết điểm, nhưng với đà phát triển mạnh mẽ của mình, thì Swift vẫn là một ngôn ngữ rất đáng để các bạn đầu tư, học tập. Ở phần tiếp theo của bài viết này, chúng tôi sẽ đề cập sâu hơn những lý do tuyệt vời mà sau khi đọc xong, chắc chắn các bạn sẽ bị hoàn toàn thuyết phục và quyết định ngay việc lựa chọn ngôn ngữ này để phát triển công việc trong tương lai của mình.
* Theo Apple, thì Swift nhanh gần gấp 3 lần Obj-C và nhanh hơn 8 lần so với ngôn ngữ Python. Không những nhanh, mà Swift còn rất mạnh mẽ, giúp cho các nhà lập trình có thể viết được những đoạn mã nâng cao, phức tạp hơn như: generics, closures, tuples, multiple returns, iterators, các mẫu lập trình hàm có sẵn và nhiều hơn thế nữa. Bên cạnh đó, Swift bảo mật và an toàn hơn Objective-C rất nhiều
* **Chúng được sử dụng miễn phí và là mã nguồn mở.** Điều này sẽ giúp cho cộng đồng Swift ngày càng phát triển hơn, người dùng có thể tìm kiếm các lỗ hổng bảo mật, cập nhật các bản vá lỗi, các tính năng, các plugin cải tiến của ngôn ngữ cho cộng đồng.

### Tìm hiểu về SwiftUI

* **Swift UI** là một framework tương tác và mang tính cách mạng, các developer của Apple phát triển nó để thiết kế ứng dụng cho cả **iOS**, **MacOS** và **tvOS**. Framework này đã được đưa ra như là một sự cập nhật và bổ sung đáng được chờ đợi nhất của ngôn ngữ lập trình **Swift** – cung cấp cho các nhà phát triển một giải pháp để có thể mang ứng dụng **iPhone** của họ sang **Mac** và các nền tảng khác của Apple một cách tiết kiệm thời gian và tài nguyên cần thiết.
* Framework này cũng có một tùy chọn để trải nghiệm các tính năng độc đáo như tốc độ liên quan với chuột và bàn phím, các tính năng đặc biệt như Touch Bar và hơn thế nữa – Trong tương lai nó chính là thứ sẽ cạnh tranh trực tiếp với Framework UI Flutter của Google và React của Facebook, là những framework dùng để phát triển ứng dụng đa nền tảng.
* Bây giờ, trong khi bắt đầu khởi chạy phát triển Swift UI và mang ngôn ngữ lập trình Swift vào thị trường đa nền tảng thì nó cũng được trang bị vô số tính năng khác, khiến các nhà phát triển ứng dụng iOS rất quan tâm đến việc sử dụng SwiftUI trong quá trình phát triển của họ.

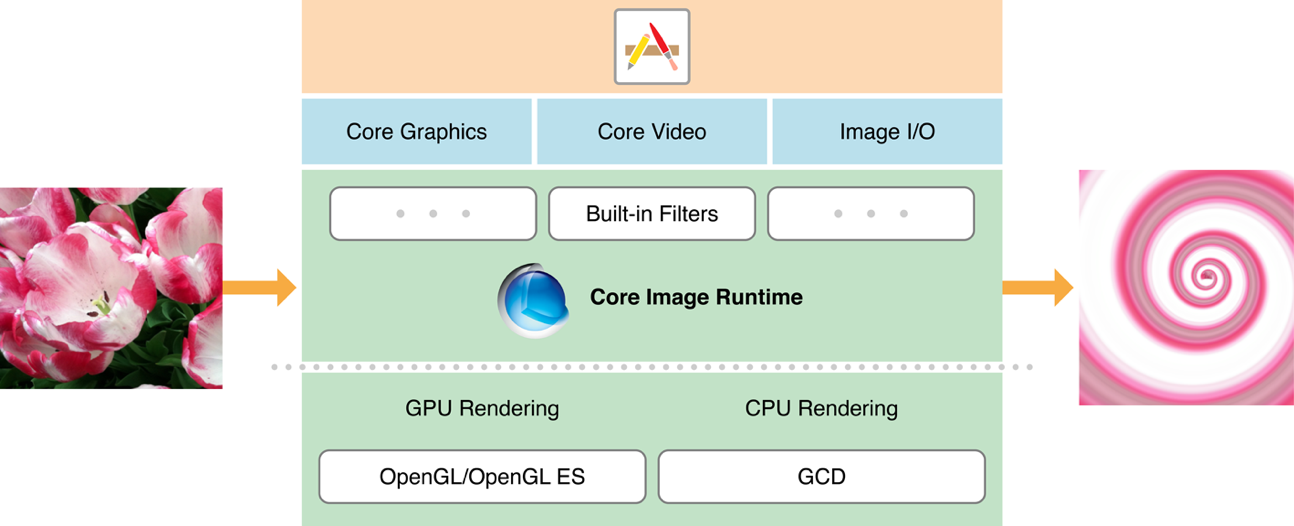
## Thư viện hỗ trợ dự án

### SwiftUI:

* SwiftUI là một UI Framework mới được phát triển bởi Apple. Nó cung cấp một API mới cho phép người dùng có thể viết Code và nó còn có thể làm việc được trên tất cả các Apple platforms: iOS, iPadOS, watchOS và tvOS!
* Mô hình này được ứng dụng từ trước iOS 13 và bây giờ đang trở thành một xu hướng mới, nó đem lại sự đơn giản trong quá trình Code và tốc độ xây dựng ứng dụng hơn so với UIKit, mặc dù vẫn chưa đủ khả năng thay thế hoàn toàn được UIKit, nhưng chưa có gì là chắc chắn cả, giống như Swift ra đời cũng chưa thay thế được Objective - C chẳng hạn.
* Nếu so sánh SwiftUI với UIKit ở thời điểm hiện tại thì đang cách biệt lớn bởi vì vẫn chưa có nhiều những sự trợ giúp từ các trang nổi tiếng như Stackoverflow, Medium, Hackingswift, ... về những rắc rối hay những vấn đề phức tạp.
* Hiện tại SwiftUI đang được nhiều người lựa chọn sử dụng mặc dù vẫn đang trong quá trình nâng cấp và bổ xung, bởi vì họ tin vào SwiftUI sẽ phát triển mạnh mẽ trong tương lai, và có một điều rằng bạn có thể sử dụng đồng thời cả UIKit và SwiftUI trong một project, vậy nó nói lên rằng nếu bạn đã thông thạo UIKit thì có thêm một sự trợ gúp hoàn hảo khi sử dụng SwiftUI. Có thể nó sẽ là một Framework mạnh mẽ trong tương lai gần, tuy còn nhiều thiếu xót nhưng Apple đã và đang ngày một hoàn thiện cho nó.

### Core Image:

* Core Image là một framework được viết bởi Apple sử dụng cho cả iOS và OS x. Đây là một framework rất mạnh mẽ, cung cấp cho chúng ta khả năng xử lý ảnh cực nhanh (near real-time processing). Framework này có khả năng xử lý nhiểu kiểu dữ liệu hình ảnh từ các framework khác như Core Graphics, Core Video, Image I/O, ...
* Core Image cung cấp cho chúng ta các hàm API đơn giản để chúng ta làm việc với tác vụ xử lý đồ họa. Thông qua các hàm API này, chúng ta không cần phải làm việc trực tiếp với các tầng xử lý đồ họa bậc thấp bên dưới (OpenGL, Open GLES). Hơn nữa, Core Image có khả năng chọn lựa xử lý hình ảnh trên CPU hoặc GPU, cung cấp cho chúng ta API để thực hiện việc xử lý đa luồng.



Hình 2.1 - Quá trình Core Image xử lý ảnh

* Khung Core Image cung cấp:
* Truy cập vào các bộ lọc xử lý hình ảnh tích hợp sẵn
* Khả năng phát hiện tính năng
* Hỗ trợ nâng cao hình ảnh tự động
* Khả năng liên kết nhiều bộ lọc với nhau để tạo hiệu ứng tùy chỉnh
* Hỗ trợ tạo các bộ lọc tùy chỉnh chạy trên GPU
* Khả năng xử lý hình ảnh dựa trên phản hồ
* Core Image cung cấp hàng trăm bộ lọc tích hợp. Bạn thiết lập bộ lọc bằng cách cung cấp các cặp khóa - giá trị cho các thông số đầu vào của bộ lọc. Đầu ra của một bộ lọc này có thể là đầu vào của bộ lọc khác, giúp bạn có thể liên kết nhiều bộ lọc với nhau để tạo ra các hiệu ứng tuyệt vời. Nếu bạn tạo một hiệu ứng ghép mà bạn muốn sử dụng lại, bạn có thể phân lớp CIFilter để nắm bắt “công thức” của hiệu ứng.
* Có hơn một chục loại bộ lọc. Một số được thiết kế để đạt được kết quả nghệ thuật, chẳng hạn như danh mục bộ lọc kiểu cách và bán sắc. Những thứ khác là tối ưu để khắc phục các vấn đề về hình ảnh, chẳng hạn như điều chỉnh màu sắc và làm sắc nét các bộ lọc.
* Core Image có thể phân tích chất lượng của hình ảnh và cung cấp một tập hợp các bộ lọc với các cài đặt tối ưu để điều chỉnh những thứ như màu sắc, độ tương phản và tông màu cũng như để chỉnh sửa các hiện tượng đèn flash, chẳng hạn như mắt đỏ. Nó thực hiện tất cả điều này với một cuộc gọi phương thức từ phía bạn.
* Core Image có thể phát hiện các đặc điểm khuôn mặt người trong ảnh tĩnh và theo dõi chúng theo thời gian trong ảnh video. Biết vị trí của các khuôn mặt có thể giúp bạn xác định vị trí đặt họa tiết hoặc áp dụng các bộ lọc đặc biệt khác.

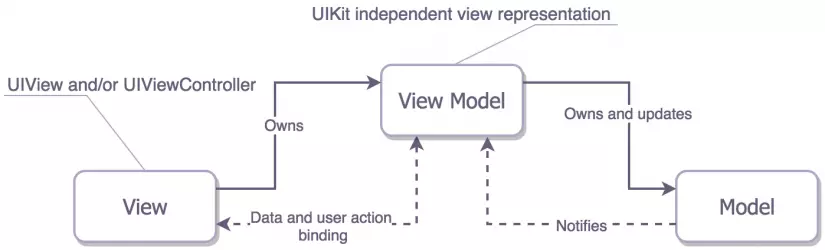
### PencilKit

* PencilKit sẽ cung cấp 1 canvas view, 1 tool picker (với 1 vài công cụ vẽ hữu dụng) như là đối tượng PKDrawing chẳng hạn, nó sẽ giúp bạn lưu, tái hiện lại các bản vẽ. Dĩ nhiên là tất cả các chức năng này phải được tự thực hiện. Khi làm việc với PencilKit, sẽ có rất nhiều điểm cần phải chú ý để làm được 1 ứng dụng hoàn chỉnh. Tuy nhiên, PencilKit lại rất tuyệt vời, nó hoạt động với độ chính xác cao, độ trễ thấp khi được sử dụng cùng vs Apple Pencil. Với PencilKit, thay vì cố phát minh lại bánh xe thì bạn có thể tập trung vào các tính năng độc đáo của ứng dụng
* Tuy có rất nhiều tính năng hữu ích, bạn đôi khi vẫn cảm thấy không được thuận tiện khi tuỳ chỉnh một vài đối tượng. Sau đây là 1 số vấn đềcó thể gặp phải:
* Rất khó để customize các tool picker
* Không thế lấy về danh sách các nét vẽ mà bản vẽ có
* Chỉ có thể lấy về các dữ liệu đại diện cho bản vẽ mà thôi

## Kiến trúc tổng quan

### Giới thiệu mô hình MVVM

* MVVM là viết tắt của Model - View - ViewModel, đây là mô hình hỗ trợ two-way data binding giữa View và View Model. Cụ thể mô hình MVVM được trình bày như sau:
* View: Tương tự như trong mô hình MVC, View là phần giao diện của ứng dụng để hiển thị dữ liệu và nhận tương tác của người dùng. Một điểm khác biệt so với các ứng dụng truyền thống là View trong mô hình này tích cực hơn. Nó có khả năng thực hiện các hành vi và phản hồi lại người dùng thông qua tính năng binding, command.
* Model: Cũng tương tự như trong mô hình MVC. Model là các đối tượng giúp truy xuất và thao tác trên dữ liệu thực sự.
* ViewModel: Lớp trung gian giữa View và Model. ViewModel có thể được xem là thành phần thay thế cho Controller trong mô hình MVC. Nó chứa các mã lệnh cần thiết để thực hiện data binding, command.
* Trong các mô hình truyền thống, chúng ta thường xử lý sự kiện Click và viết mã thực thi trực tiếp ở trên một Button nhưng với mô hình MVVM không cho phép làm điều này.
* Trong mô hình MVVM, các điều khiển (control) như Button, ListView, SearchBar, ... không thể kết buộc trực tiếp đến dữ liệu mà phải thông qua thuộc tính Command – là một thuộc tính kiểu ICommand.
* Một điểm cần lưu ý là trong mô hình MVVM, các tầng bên dưới sẽ không biết được các thông tin gì về tầng bên trên nó. Như hình minh họa dưới đây:



Hình 2.2 - Các tầng mô hình MVVM

### Ưu nhược điểm của MVVM

* Ưu điểm:
* Thực hiện Unit testing bây giờ sẽ rất dễ dàng, vì bạn thực sự không phụ thuộc vào view.
* MVVM sẽ tạo sự tương tác hiệu quả giữa designer và developer.
* Tăng khả năng sử dụng lại các thành phần hay việc thay đổi giao diện chương trình mà không cần phải viết lại code quá nhiều.
* Phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì…
* Nhược điểm :
* Khả năng duy trì khi view có thể gán cả biến và biểu thức, các logic không liên quan sẽ tăng dần theo thời gian, ảnh hưởng đến việc thêm code vào XML.
* Đối với dự án nhỏ việc áp dụng mô hình MVVM gây cồng kềnh, tốn thời gian trong quá trình phát triển. Tốn thời gian trung chuyển dữ liệu của các thành phần.
* Đối với dự án lớn hơn, nó gây khó khăn và mất thời gian để thiết kế các ViewModel.
* Việc liên kết dữ liệu cho tất cả các thành phần gây khó khăn trong việc debug khi cơ sở dữ liệu phức tạp.

# PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## Giới thiệu Xcode

* Xcode là một ứng dụng được Apple xây dựng để hỗ trợ cho các lập trình viên phát triển các ứng dụng chạy trên các nền tảng của hãng (iOS, macOS…). Xcode là một công cụ cực kỳ hữu dụng, rất tuyệt vời… và dân lập trình hay ưu ái gọi rằng nó rất mạnh.
* Thực sự vậy, Xcode là công cụ vô cùng hữu ích để xây dựng các ứng dụng trên nền tảng của Apple. Với các lập trình viên ứng dụng nếu đã sử dụng Xcode thì sẽ thấy được sự hỗ trợ rất mạnh mẽ của nó, từ việc viết code đến debug chức năng, tìm lỗi. Tất cả được hỗ trợ vô cùng tuyệt vời mà ít công cụ nào làm được. Tính đến thời điểm hiện tại phiên bản Xcode là 13.4
* Giao diện làm việc của phần mềm Xcode gồm có 5 part chính: **Toolbar**, **Editor area**, **Navigator area**, **Debug area**, **Utility area**.
* Debug area: là vùng hỗ trợ lập trình viên trong quá trình debug lỗi của phần mềm.
* Toolbar area: đây là vùng chứa các công cụ tiện ích giúp lập trình viên có thể đơn giản hơn trong việc chạy, debug ứng dụng, lựa chọn iOS Simulator hay đóng mở các vùng khác…
* Editor area: vùng để bạn có thể thiết kế giao diện, viết và sửa code của chương trình.
* Utility area: vùng này cho phép lập trình viên tùy chỉnh các tham số, giá trị của các đối tượng trong giao diện, cũng như cho phép người sử dụng kéo thả và sử dụng các đối tượng có sẵn của Xcode như Button, Label, Slider… hay các đoạn code mẫu (If, Switch…).
* Navigator area: cung cấp cho lập trình viên một cái nhìn trực quan, tiện lợi trong việc quản lý ứng dụng, xem thông báo lỗi, tìm kiếm một đoạn code trong chương trình hay ra soát mức độ hoạt động của RAM, CPU khi chạy ứng dụng…



Hình 3.1 - Logo phần mềm Xcode

## Xác định các tác nhân

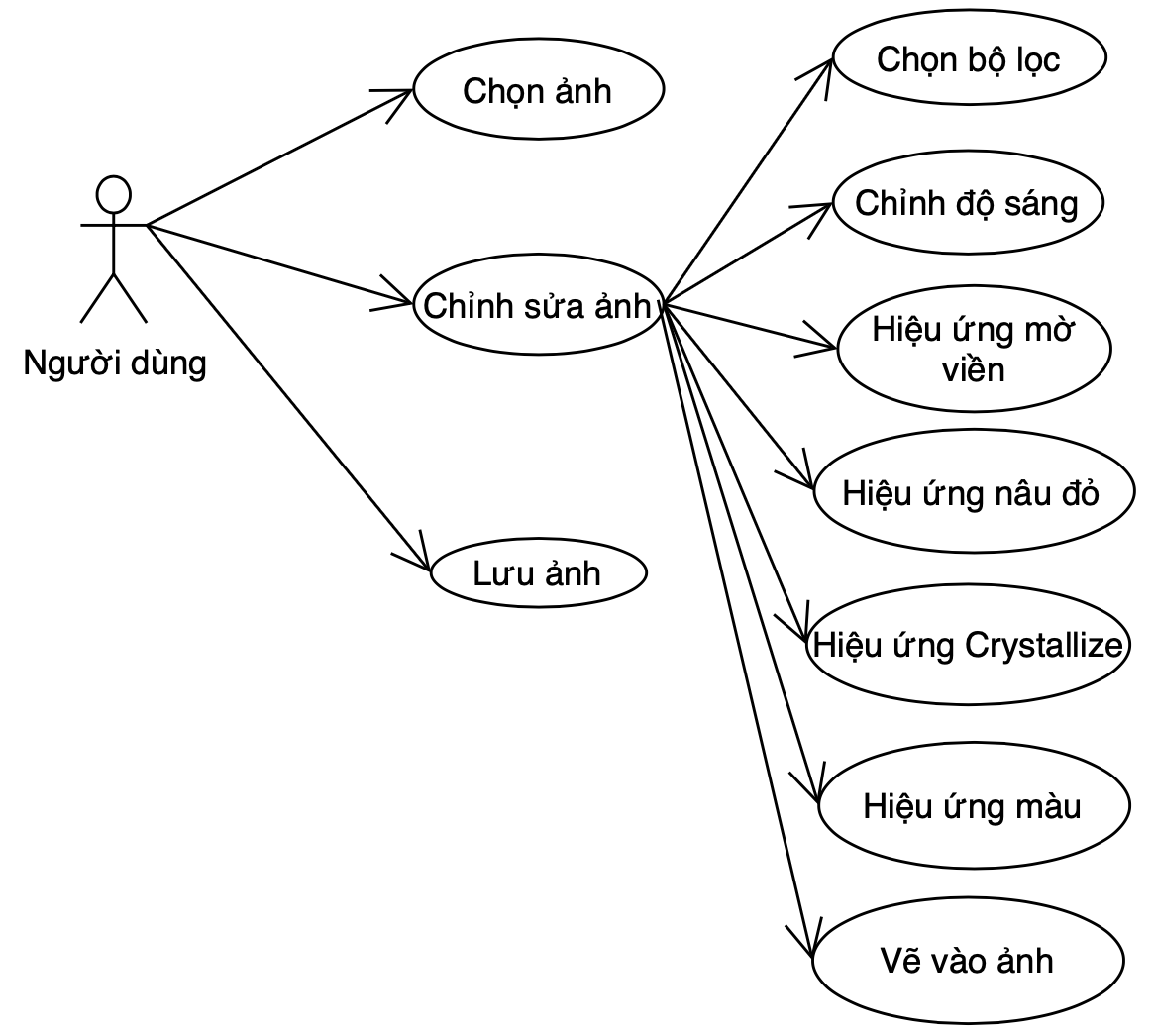
### Sơ đồ ca sử dụng hệ thống



Hình 3.2 - Sơ đồ ca sử dụng hệ thống

* Sơ đồ ca sử dụng hệ thống mô tả sự tương tác giữa người dùng và hệ thống với nhau, mối quan hệ giữa những tương tác đó trong một môi trường cụ thể và vì một mục đích cụ thể. Và với đề tài này, chỉ có một ca sử dụng là “Người dùng”.
* Sự tương tác ở đây đó là:
* Người dùng lưu ảnh.
* Người dùng mở thư viện.
* Người dùng chỉnh sửa ảnh.
* Mục đích:
* Người dùng có thể chụp hình sau đó lưu ảnh lại vào thư viện ảnh.
* Mở thư viện ảnh, chỉnh sửa ảnh và lưu ảnh.

### Sơ đồ ca sử dụng cho người dùng



Hình 3.3 - Sơ đồ ca sử dụng cho người dùng

* Tương tự như sơ đồ ca sử dụng cho hệ thống, nhưng với sơ đồ cho người dùng, thì nó đơn giản hơn, mô tả các chức năng mà người dùng có thể tương tác với hệ thống.
* Tác nhân cũng là “Người dùng”, với các sự tương tác: Chọn ảnh, lưu hình, chỉnh sửa hình

### Mô tả ca sử dụng

Bảng 3.1 - Chức năng chỉnh ảnh theo bộ lọc

|  |  |
| --- | --- |
| **Use case** | **Nội dung** |
| Tên Use case | Chỉnh ảnh theo bộ lọc |
| Tác nhân | Người dùng |
| Mô tả | Chọn ảnh từ thư viện (nhấn chọn icon ảnh trên màn hình) và thực hiện chỉnh sửa ảnh theo bộ lọc có sẵn |
| Dữ liệu vào | Hình ảnh từ thư viện |
| Kết quả | Hình được chỉnh sửa và có được một ảnh mới |
| Quá trình xử lý | 1. Chọn ảnh từ thư viện ảnh. 2. Nhấn chọn nút Filter. 3. Chọn filter thích hợp. 4. Tiếp tục bước chỉnh sửa hoặc nhấn lưu. 5. Kết thúc Use case. |

Bảng 3.2 - Chức năng chỉnh ảnh tuỳ chỉnh

|  |  |
| --- | --- |
| **Use case** | **Nội dung** |
| Tên Use case | Chỉnh ảnh tuỳ chỉnh |
| Tác nhân | Người dùng |
| Mô tả | Chọn ảnh từ thư viện (nhấn chọn icon ảnh trên màn hình) và thực hiện chỉnh sửa ảnh theo ý muốn với các tuỳ chọn: Brightness, Vignette, Sepia, Crystallization, Color |
| Dữ liệu vào | Hình ảnh từ thư viện |
| Kết quả | Hình được chỉnh sửa và có được một ảnh mới |
| Quá trình xử lý | 1. Chọn ảnh từ thư viện ảnh. 2. Nhấn chọn nút Edit. 3. Kéo thanh seekbar đến vị trí cần chọn để có hiệu ứng thích hợp. 4. Tiếp tục bước chỉnh sửa hoặc nhấn lưu. 5. Kết thúc Use case. |

Bảng 3.3 - Chức năng vẽ vào ảnh

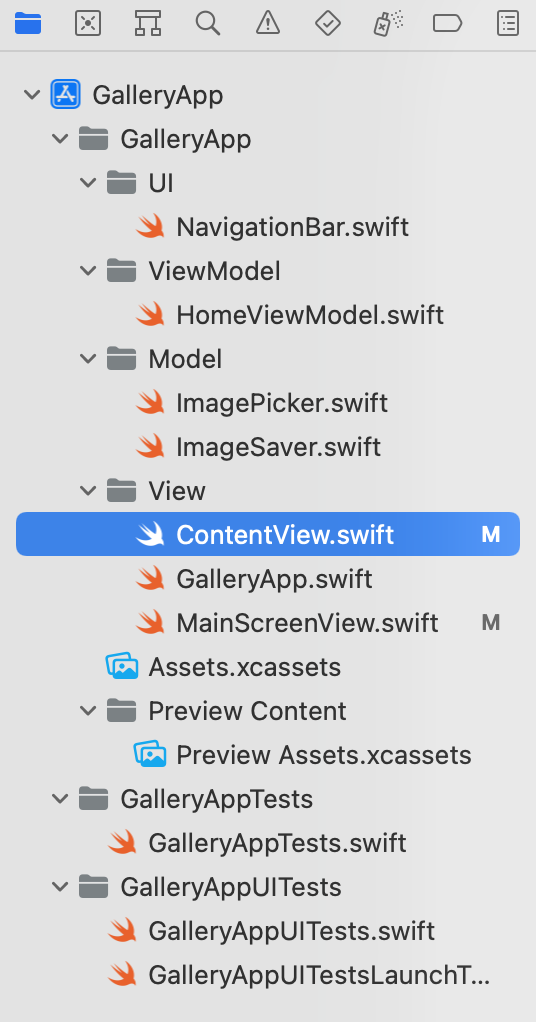
|  |  |
| --- | --- |
| **Use case** | **Nội dung** |
| Tên Use case | Vẽ vào ảnh |
| Tác nhân | Người dùng |
| Mô tả | Chọn ảnh từ thư viện (nhấn chọn icon ảnh trên màn hình) và thực hiện tuỳ chỉnh nét vẽ, màu và vẽ lên ảnh |
| Dữ liệu vào | Hình ảnh từ thư viện |
| Kết quả | Hình được chỉnh sửa và có được một ảnh mới |
| Quá trình xử lý | 1. Chọn ảnh từ thư viện ảnh. 2. Nhấn chọn nút Brush. 3. Kéo thanh seekbar để chọn độ rộng của nét vẽ, và chọn màu. 4. Tiếp tục bước chỉnh sửa hoặc nhấn lưu. 5. Kết thúc Use case. |

## Triển khai xây dựng

### Nhập thư viện

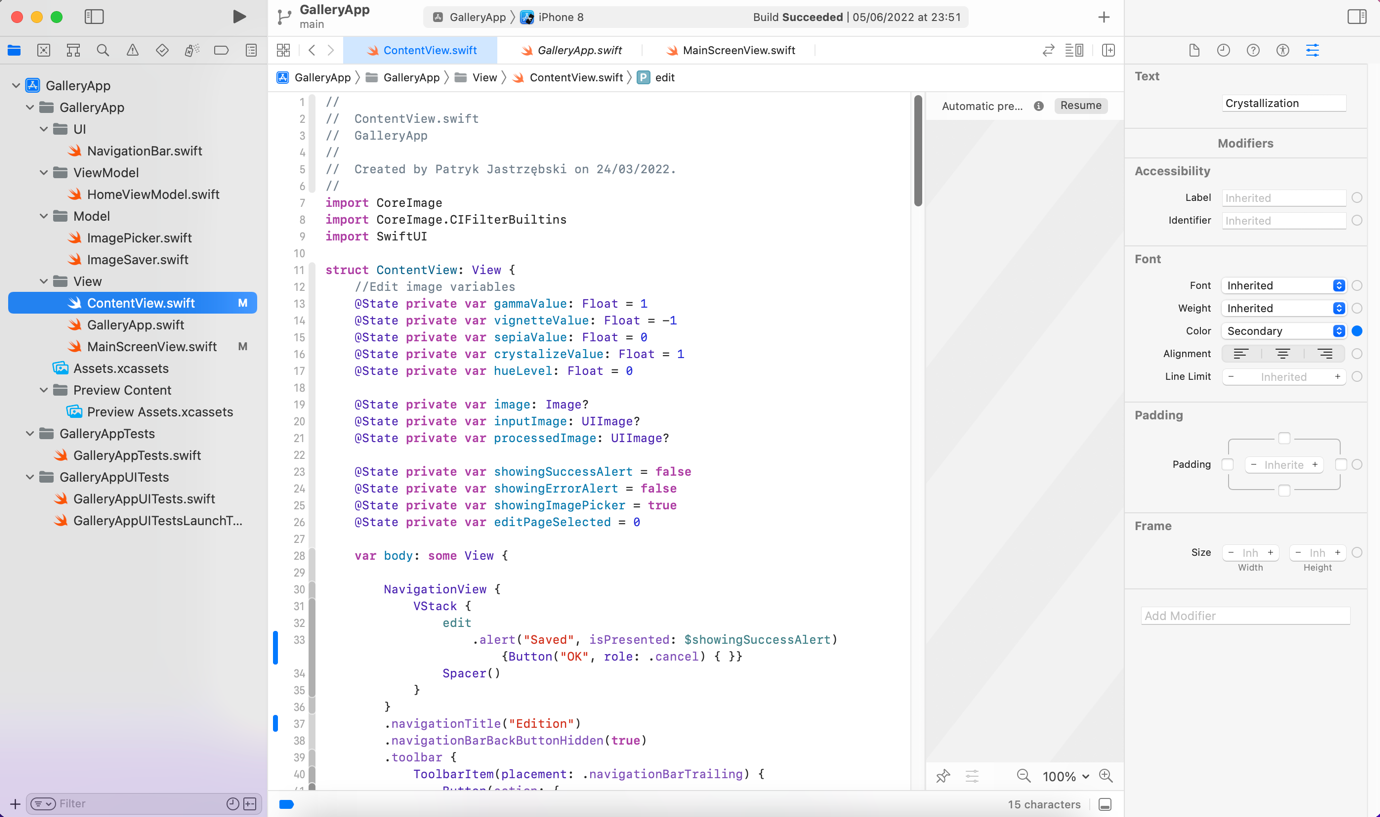
* Trước khi bạn bắt đầu, hãy nhập một số thư viện quan trọng nhất trong khung Core Image:
* CIContext: **CIContext** thực hiện tất cả các xử lý của một hình ảnh cốt lõi. Điều này giống như một ngữ cảnh Core Graphics hoặc OpenGL.
* CIImage: Lớp này chứa dữ liệu hình ảnh. A **UIImage**, một tệp hình ảnh hoặc dữ liệu pixel có thể tạo ra nó.
* CIFilter: Lớp **CIFilter** học có từ điển. Điều này xác định các thuộc tính của bộ lọc cụ thể mà nó đại diện. Ví dụ về bộ lọc bao gồm sự sống động, đảo màu, cắt xén và nhiều hơn nữa.
* SwiftUI: tạo giao diện cho ứng dụng
* PencilKit: cung cấp nhiều chức năng vẽ

### Cây thư mục Project



Hình 3.4 - Cây thư mục Project

### Code giao diện chính



Hình 3.5 - Code giao diện chính

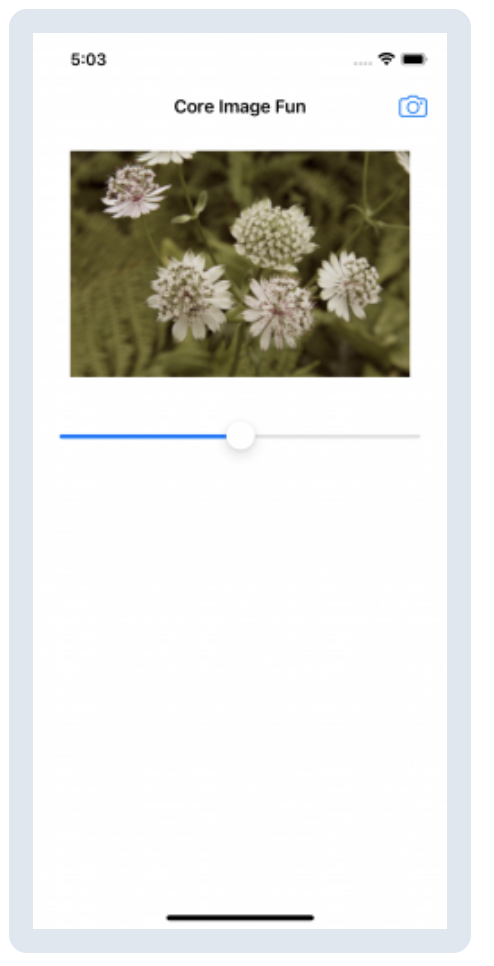
## Phân tích thuật toán

### Áp dụng bộ lọc

* Diễn giải thuật toán:
* Tạo một đối tượng CIImage: **CIImage** có một số phương thức khởi tạo. Trong hướng dẫn này, bạn sẽ sử dụng **CIImage(image:)** để tạo **CIImage** từ **UIImage**.
* Tạo CIContext: **CIContext** có thể dựa trên CPU hoặc GPU. **CIContext** rất tốn kém để khởi tạo, vì vậy bạn sử dụng lại nó thay vì tạo nó nhiều lần. Bạn sẽ luôn cần một cái khi xuất **CIImage** đối tượng.
* Tạo CIFilter: Khi bạn tạo bộ lọc, bạn định cấu hình một số thuộc tính trên nó phụ thuộc vào bộ lọc bạn đang sử dụng.
* Nhận đầu ra bộ lọc: Bộ lọc cung cấp cho bạn hình ảnh đầu ra dưới dạng **CIImage**. Bạn có thể chuyển đổi này thành một **UIImage** bằng cách sử dụng **CIContext**
* Tạo giao diện người dùng và sử dụng nó để tạo **CIImage**.
* Tạo **CIFilter** loại **CISepiaTone**. Đó là loại sepia-tone.
* Bộ lọc **CISepiaTone** có hai giá trị. Đầu tiên, một hình ảnh đầu vào: **kCIInputImageKey**, là một phiên bản **CIImage**. Thứ hai, cường độ: **kCIInputIntensityKey**, một giá trị nổi giữa 0 và 1. Hầu hết các bộ lọc sử dụng các giá trị mặc định của chúng nếu không có bất kỳ giá trị nào. Một ngoại lệ là **CIImage**. Điều này phải cung cấp một giá trị vì không có mặc định.
* Lấy **CIImage** trở lại từ bộ lọc, sử dụng thuộc tính **outputImage**.
* Chuyển **CIImage** trở lại giao diện người dùng và hiển thị nó trong chế độ xem hình ảnh.
* Tiếp theo, hãy gọi phương thức mới được thêm vào bằng cách thêm các phương thức sau vào **viewDidLoad()**:

|  |
| --- |
| applySepiaFilter(intensity: 0.5) |

* Điều này kích hoạt lọc hình ảnh với giá trị cường độ là 0,5. Ở phần sau của hướng dẫn, bạn sẽ sử dụng thanh trượt để thử các giá trị cường độ khác nhau.
* Xây dựng và chạy dự án. Ta sẽ thấy hình ảnh của mình được lọc bởi bộ lọc tông màu nâu đỏ:



Hình 3.6 - Kết quả sau khi áp dụng CIImage và CIFilter

### Thay đổi giá trị bộ lọc

* Đã có một **IBAction** kết nối với hành động Thay đổi Giá trị của thanh trượt. Nó được gọi là **sliderValueChanged(\_:)**. Trong phương pháp này, bạn sẽ làm lại bộ lọc hình ảnh bất cứ khi nào giá trị thanh trượt thay đổi. Nhưng bạn không muốn làm lại toàn bộ quá trình. Điều đó sẽ khá không hiệu quả và sẽ mất quá nhiều thời gian. Giữ một số đối tượng bạn tạo trong áp dụng hàm **applySepiaFilter(intensity:)**.
* Thêm các thuộc tính sau ngay bên dưới tờ khai **context**:

|  |
| --- |
| let filter = CIFilter(name: "CISepiaTone")! |

* Tiếp theo, thêm những điều sau đây vào **viewDidLoad()** trước khi gọi **applySepiaFilter(intensity:)** :

|  |
| --- |
| guard let uiImage = UIImage(named: "image") else { return }  let ciImage = CIImage(image: uiImage)  filter.setValue(ciImage, forKey: kCIInputImageKey) |

* Di chuyển nó đến **viewDidLoad()** để ngăn cuộc gọi trên mọi thay đổi giá trị thanh trượt, sau đó thay thế áp dụng **applySepiaFilter(intensity:)** như sau:

|  |
| --- |
| func applySepiaFilter(intensity: Float) {  filter.setValue(intensity, forKey: kCIInputIntensityKey)  guard let outputImage = filter.outputImage else { return }  guard let cgImage = context.createCGImage(outputImage, from: outputImage.extent) else { return }  imageView.image = UIImage(cgImage)  } |

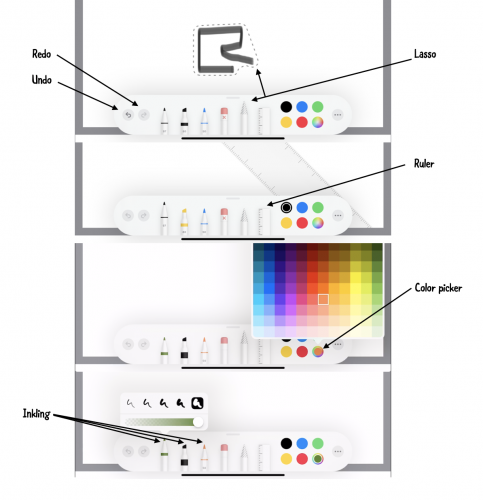
* Cuối cùng thêm hàm **sliderValueChanged(\_:)** :

|  |
| --- |
| applySepiaFilter(intensity: slider.value) |

* Khi giá trị thanh trượt thay đổi, **applySepiaFilter(intensity:)** sẽ chạy với giá trị cường độ mới. Thanh trượt của bạn được đặt thành các giá trị mặc định: tối thiểu 0, tối đa 1, mặc định 0,5.

### Vẽ vào ảnh

* Diễn giải thuật toán:
* PKInkingTool: Công cụ để vẽ đường. Bạn đã thấy điều này trong hành động
* PKEraserTool: Bàn chải để xóa các phần của bản vẽ của bạn
* PKLassoTool: Công cụ để chọn các phần của bản vẽ của bạn để bạn có thể đặt lại vị trí của nó
* Ruler: Giúp bạn vẽ các đường thẳng
* Color picker: Sử dụng nó để chọn màu cho công cụ mực của bạn
* Undo/Redo: Các điều khiển bạn sử dụng để hoàn tác và làm lại các thay đổi trên bản vẽ



Hình 3.7 - Các chức năng cung cấp bởi PKToolPicker

* Đầu tiên: Tạo một nút **UIBarbutton** để xóa, hoàn tác, làm lại và Chế độ xem hình ảnh trong điều hướng trong ViewController của dự án. Thêm hành động và lối thoát cho nút rõ ràng và hình ảnh từ bảng phân cảnh.
* Tạo một chức năng để thêm chế độ xem được thiết lập cho chế độ xem canvas trong View Controller

|  |
| --- |
| override func viewDidLoad() {      super.viewDidLoad()      canvasView.delegate = self      canvasView.becomeFirstResponder()  }    override func viewDidAppear(\_ animated: Bool) {      super.viewDidAppear(true)      setUpCanvasView()  }    func setUpCanvasView() {          if let window = view.window, let toolPicker = PKToolPicker.shared(for: window) {              toolPicker.setVisible(true, forFirstResponder: canvasView)              toolPicker.addObserver(canvasView)          }      } |

* Bây giờ hãy thêm chức năng đại biểu cho chế độ xem canvas trong View Controller

|  |
| --- |
| extension ViewController: PKCanvasViewDelegate {        func canvasViewDrawingDidChange(\_ canvasView: PKCanvasView) {          print("drawing")      }        func canvasViewDidEndUsingTool(\_ canvasView: PKCanvasView) {          print("End using the PK tools")      }        func canvasViewDidFinishRendering(\_ canvasView: PKCanvasView) {          print("Completed the UI Event")      }    } |

* Bây giờ thêm View Controller để xem hình ảnh của bạn được tạo sau khi vẽ trong chế độ xem canvas

|  |
| --- |
| class SecondViewController: UIViewController {      @IBOutlet weak var trashBtn: UIButton!      @IBOutlet weak var imgView: UIImageView!      var image = UIImage()      override func viewDidLoad() {          super.viewDidLoad()          imgView.image = image      }      @IBAction func trashBtnTapped(\_ sender: UIButton) {          self.dismiss(animated: true, completion: nil)      }  } |

* Bây giờ thêm điều hướng trên nút Hình ảnh trong View Controller trong dự án.

|  |
| --- |
| @IBAction func imageBtnTapped(\_ sender: UIBarButtonItem) {          let image = canvasView.asImage()          if let secodVC = self.storyboard?.instantiateViewController(identifier: "SecondViewController") as? SecondViewController {          secodVC.image = image          self.navigationController?.present(secodVC, animated: true, completion: nil)      }  } |

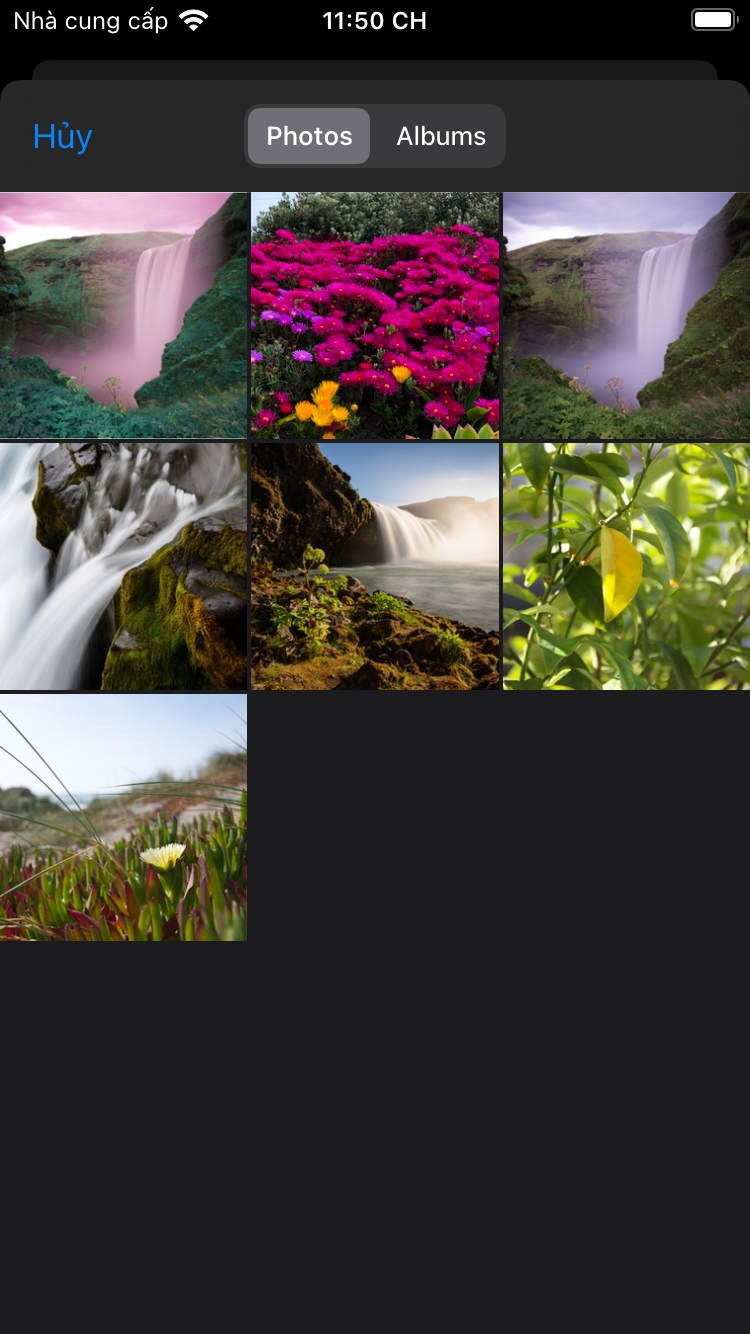
## Kết quả cài đặt

### Giao diện chính ứng dụng



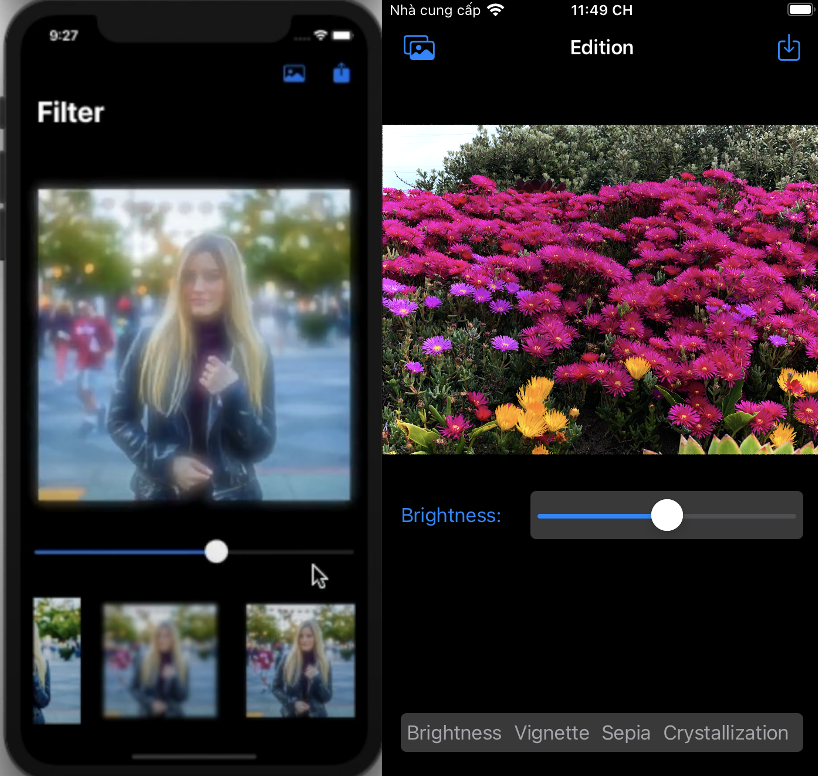
Hình 3.8 - Giao diện chính ứng dụng

### Giao diện chọn ảnh từ thư viện



Hình 3.9 - Giao diện chọn ảnh từ thư viện

### Giao diện chỉnh sửa ảnh



Hình 3.10 - Giao diện chỉnh sửa ảnh

### Giao diện vẽ vào ảnh



# 

**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

1. **Kết quả đạt được**

* Ngày nay, công đoạn chỉnh sửa này đã trở thành một phần không thể thiếu, đặc biệt khi ngành thương mại điện tử ngày càng nở rộ hiện nay. Khách hàng không thể chạm vào sản phẩm như trong cuộc sống thực, họ cần tới những bức ảnh hiển thị sản phẩm rõ ràng và sắc nét nhất. Để đạt được yêu cầu này, các hình ảnh cần nhiều thời gian và nỗ lực để chỉnh sửa hình ảnh sản phẩm của mình. Bởi họ hiểu rằng hình ảnh chuyên nghiệp bao giờ cũng tạo ấn tượng tốt cho thương hiệu của mình, giúp khách hàng vượt ra ranh giới của sự đắn đo giữ việc xem tiếp hay tắt màn hình, nên mua hay không nên mua. Và như mọi người đã biết, trước khi có mạng truyền thông xã hội, việc mọi người tự chụp ảnh ở nhà cho người khác xem thực sự là một điều vô lý. Nhưng bây giờ với Facebook, Instagram và nhiều ứng dụng khác thì ảnh “tự sướng” trở thành một sản phẩm đại chúng – từ thanh thiếu niên đến ông bà cụ đều có thể đăng ảnh của chính mình. Mỗi chia sẻ như vậy sẽ mang hình ảnh cá nhân về cuộc sống của bạn vào cuộc đua trực tuyến cùng mọi người. Bạn không muốn bị bỏ lại sau lưng, hoặc tệ hơn, bạn không muốn người người khác bình phẩm về ngoại hình hay những việc bạn làm. Những cảm giác này thúc đẩy bạn chọn đăng những hình ảnh tốt nhất của bản thân, “trang hoàng” ngoại hình bằng các ứng dụng. Và bạn sẽ có cảm giác mình chưa đủ ấn tượng nếu chưa giống những người có ảnh hưởng mà bạn theo dõi.
* Những thao tác tưởng chừng như vụn vặt này đều đóng vai trò quan trọng để tạo ra một bức hình hoàn hảo. Thông thường, người chụp hình sẽ tiến hành trên nhiều góc chụp trước khi chọn được góc nhìn ưng ý nhất. Tuy nhiên, không phải bức hình nào cũng đẹp và việc sử dụng các ứng dụng cắt chỉnh ảnh là thực sự cần thiết. Sau 2 tháng tìm hiểu, thực hiện đề tài cũng như viết báo cáo về đề tài “Xây dựng ứng dụng chỉnh sửa ảnh trên iOS” đến nay tôi đã đạt được một số kết quả như sau: Chụp hình, chỉnh sửa ảnh theo Filter, chèn ảnh, icon, và các hình vẽ: cắt chỉnh ảnh theo ý muốn. Với giao diện thân thiện và dễ sử dụng, các tính năng khá phổ biến chúng tôi mong rằng có thể phát triển đề tài này cho những đồ án sau này.
* Với đồ án lần này, ngoài việc xây dựng một ứng dụng có thể chụp và chỉnh sửa ảnh, tôi còn được rèn luyện kĩ năng code trên Xcode, biết thêm về các hàm xử lí ảnh, và hệ màu trong ngôn ngữ lập trình Swift. Nâng cao khả năng lập trình, tự học hỏi nghiên cứu tài liệu, kĩ năng làm báo cáo, cũng như làm nhóm. Cùng với sự giúp đỡ của ThS. Trịnh Thị Ngọc Linh, nhóm chúng tôi đã có thể hoàn thành tốt đồ án lần này.

1. **Hạn chế**

Tuy đã đạt được những chức năng đề ra ban đầu nhưng ứng dụng vẫn còn rất nhiều hạn chế:

* Các chức năng chỉnh sửa chưa thực sự hoàn thiện: chạy còn chậm
* Các tính năng chưa tối ưu, giao diện còn đơn giản
* Chưa có tính năng mới tạo nên sự khác biệt với các ứng dụng chỉnh sửa ảnh hiện nay
* Các chức năng phổ biến và đơn giản

1. **Hướng phát triển**

Ứng dụng vẫn còn thiếu sót khá nhiều về mặt chức năng cũng như khả năng tương tác với người dùng, do đó trong thời gian tới sẽ tiến hành cập nhập bổ sung các tính năng sau:

* Tiến hành nâng cấp giao diện của ứng dụng.
* Thêm các chức năng như đảo ảnh, lọc mịn ảnh, làm đẹp (trang điểm, nhận diện, chỉnh sửa khuôn mặt)
* Có thể chụp hình cùng Filter.
* Cắt background

**DANH MỤC TÀI LIỆU THAM KHẢO**

[1] “<https://www.raywenderlich.com/25658084-core-image-tutorial-for-ios-custom-filters>”

[2] “<https://www.raywenderlich.com/30195423-core-image-tutorial-getting-started>”

[3] “<https://kavsoft.dev/>”

[4] “<https://www.hackingwithswift.com/learn>”

[5] “<https://mobikul.com/pencilkit-in-swift/>”