TRƯỜNG ĐẠI HỌC LẠC HỒNG

KHOA CÔNG NGHỆ THÔNG TIN

-----ꕥ📖ꕥ-----

BÁO CÁO
NGHIÊN CỨU KHOA HỌC

ĐỀ TÀI:

XÂY DỰNG ỨNG DỤNG LUYỆN THI LÝ THUYẾT LÁI XE TRÊN HỆ ĐIỀU HÀNH IOS (DÙNG CHO IPHONE, IPAD)

PHẠM THỊ NINH KIỀU

BIÊN HÒA, THÁNG 12/2014

TRƯỜNG ĐẠI HỌC LẠC HỒNG

KHOA CÔNG NGHỆ THÔNG TIN

-----ꕥ📖ꕥ-----

BÁO CÁO
NGHIÊN CỨU KHOA HỌC

ĐỀ TÀI:

XÂY DỰNG ỨNG DỤNG LUYỆN THI LÝ THUYẾT LÁI XE TRÊN HỆ ĐIỀU HÀNH IOS (DÙNG CHO IPHONE, IPAD)

SVTH : PHẠM THỊ NINH KIỀU

GVHD : PGS.TS TRẦN VĂN LĂNG

BIÊN HÒA, THÁNG 12/2014

LỜI CÁM ƠN

Để hoàn thành đề tài và có được kiến thức như ngày hôm nay, đầu tiên em xin gửi lời cảm ơn đến Ban Giám Hiệu cùng toàn thể Thầy Cô Khoa Công Nghệ Thông Tin – Trường Đại Học Lạc Hồng đã tận tình giảng dạy, truyền đạt kiến thức cũng như những kinh nghiệm quý báu cho em trong suốt quá trình học tập tại trường.

Em xin gửi những lời tri ân sâu sắc nhất đến PGS.TS Trần Văn Lăng, người thầy đã tận tình hướng dẫn và quan tâm, giúp đỡ, động viên em trong suốt quá trình thực hiện đề tài.

Em vô cùng cảm ơn sự nhiệt tình của thầy Tạ Nguyễn, anh Cao Thanh Vàng, các anh, chị và các bạn đã hỗ trợ em trong quá trình nghiên cứu, tìm hiểu.

Em cũng bày tỏ lòng biết ơn đến những người thân trong gia đình đã động viên và tạo mọi điều kiện giúp em trong quá trình học tập cũng như trong cuộc sống.

Mặc dù đã cố gắng hoàn thành tốt đề tài nhưng cũng không thể tránh khỏi những sai sót hạn chế nhất định, rất mong được sự thông cảm và chia sẻ cùng quý Thầy Cô và bạn bè.

Em xin gửi lời chúc sức khỏe và thành đạt tới tất cả quý thầy cô cùng các bạn.

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Biên Hòa, Ngày … tháng … năm 2014

Kí tên

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC HÌNH

PHẦN MỞ ĐẦU

1. Tên đề tài

"XÂY DỰNG ỨNG DỤNG LUYỆN THI LÝ THUYẾT LÁI XE TRÊN HỆ ĐIỀU HÀNH IOS (DÙNG CHO IPHONE, IPAD)".

2. Lý do chọn đề tài

Hiện nay khi nhu cầu đi lại, vận chuyển ngày càng gia tăng, đặc biệt là tại các thành phố lớn, các khu công nghiệp, những nơi tập trung đông dân cư thì vấn đề an toàn giao thông là một vấn đề bức thiết mà toàn xã hội quan tâm.

Theo thống kê của Ủy ban An toàn giao thông Quốc gia, trong 9 tháng năm 2014, toàn quốc xảy ra 18.697 vụ tai nạn giao thông, làm chết 6.758 người, bị thương 17.835 người. So với cùng kỳ năm 2013, giảm cả ba tiêu chí với 3.164 vụ tai nạn, 282 người chết và 3.945 người bị thương. Mặc dù số vụ, số người bị thương, số người chết có giảm so với cùng kỳ năm 2013, tuy nhiên tai nạn giao thông vẫn ở mức cao và còn xảy ra một số vụ đặc biệt nghiêm trọng liên quan đến xe khách, xe ô tô.[18]

Cũng theo đánh giá của Uỷ ban An toàn giao thông Quốc gia, nguyên nhân của tình trạng trên là do công tác tuần tra, kiểm soát, xử lý vi phạm chỉ mới tập trung tại một số địa bàn, một số nhóm đối tượng trên các tuyến đường trọng điểm, trong giờ cao điểm, chưa thường xuyên và liên tục. Bên cạnh đó, công tác quản lý về hoạt động vận tải, đào tạo, sát hạch cấp giấy phép lái xe, đăng kiểm còn một số hạn chế, tình trạng phương tiện chở quá tải trọng, hiện tượng tiêu cực trong quá trình thực thi nhiệm vụ vẫn còn xảy ra.

Ngoài ra, việc vi phạm pháp luật về giao thông cũng là một nguyên nhân dẫn đến tai nạn giao thông vì vậy việc nâng cao nhận thức, ý thức chấp hành luật giao thông là nhu cầu cần thiết. Bất kì người dân nào sử dụng phương tiện giao thông lưu thông trên đường đều cần phải có giấy phép lái xe; từ thực tế đó, các trung tâm tổ chức thi cấp giấy phép lái xe thường xuyên tổ chức các lớp học luyện thi lý thuyết và tổ chức thi lấy giấy phép lái xe. Tuy nhiên, không phải ai cũng có thời gian để đến các lớp học luyện thi lý thuyết lái xe vì vậy để giải quyết vấn đề này đã có các

phần mềm giúp ôn thi lý thuyết lái xe tại nhà mà không cần đến trung tâm. Với mục đích giúp cho người dùng iOS có thể tự ôn thi tại nhà và quan trọng nhất là khuyến khích người dùng tìm hiểu, nâng cao ý thức chấp hành luật giao thông nhằm góp phần giảm tỉ lệ tai nạn giao thông.

Không ngoài mục đích trên, đề tài "Xây dựng ứng dụng luyện thi lý thuyết lái xe trên hệ điều hành iOS (dành cho iPhone, iPad)", em mong muốn được áp dụng những kiến thức đã học ở trường cùng với việc tìm hiểu các ngôn ngữ mới và môi trường lập trình mới để xây dựng ứng dụng giúp luyện thi lý thuyết lái xe. Khi mà hiện nay việc sử dụng các thiết bị thông minh đang bùng phát mạnh mẽ, số người sử dụng smartphone, tablet tăng nhanh chóng tại Việt Nam nói riêng và trên toàn thế giới nói chung. Em mong rằng ứng dụng này sẽ giúp ích cho việc ôn thi lý thuyết lái xe một cách dễ dàng, tiện lợi và ở bất cứ nơi đâu với chỉ một chiếc iPhone hoặc iPad.

3. Lịch sử nghiên cứu

Theo tìm hiểu, hiện nay ứng dụng luyện thi lý thuyết lái xe chạy trên hệ điều hành iOS cũng có khá nhiều, điển hình là một vài ứng dụng như:

- Luyện thi bằng lái xe máy (Ngo Van Trung).
- Học luật giao thông (Bizzon – 17/12/2013).
- 450 câu lý thuyết sát hạch ô tô (Ton Nguyen – 18/09/2013).
- Luật giao thông đường bộ Việt Nam (iTechForLife – 12/11/2012).

4. Mục tiêu nghiên cứu

Tìm hiểu cách thức lập trình với ngôn ngữ Objective C.

Tìm hiểu cách thức xây dựng ứng dụng trên hệ điều hành iOS.

Tìm hiểu cách thức đưa ứng dụng lên thiết bị (iPhone, iPad).

Xây dựng ứng dụng trên iPhone, iPad giúp luyện thi những câu hỏi về lý thuyết trong việc điều khiển phương tiện lưu thông trên đường phố.

5. Đối tượng và phạm vi nghiên cứu

5.1. Đối tượng nghiên cứu

- Ngôn ngữ Objective C.

- Phần mềm Xcode.
- Bộ câu hỏi luyện thi lý thuyết lái xe.
- Các loại biển báo.

5.2. Phạm vi nghiên cứu

- Câu hỏi trắc nghiệm thi lý thuyết lái xe (xe máy, ô tô).
- Sử dụng cơ sở dữ liệu để lưu trữ.
- Đưa ứng dụng lên thiết bị iPhone, iPad.

6. Phương pháp nghiên cứu

Tìm hiểu về công cụ Xcode hỗ trợ cho việc viết ứng dụng dành cho hệ điều hành iOS.

Thu thập, tìm hiểu các tài liệu về ngôn ngữ Objective C.

Tìm hiểu các đối tượng và cách sử dụng các đối tượng trong Xcode để thiết kế ứng dụng.

Thu thập dữ liệu về việc thi lý thuyết lái xe: bộ câu hỏi, số lượng câu hỏi cho mỗi loại phương tiện, …

Tìm hiểu cách đưa ứng dụng lên thiết bị thật, cách đóng gói và chia sẻ ứng dụng sau khi hoàn tất.

7. Những đóng góp mới của đề tài và những vấn đề chưa thực hiện được

Những vấn đề chưa thực hiện được:

Việc sử dụng cách lựa chọn ngẫu nhiên các câu hỏi mà không tạo thành các dạng đề cho trước khiến cho việc bị trùng lắp và việc trả lời một hay nhiều câu hỏi quá nhiều lần, hạn chế việc thi thử toàn bộ các câu hỏi.

Chưa tập hợp được một số mẹo để hỗ trợ người dùng trong quá trình thi và ôn thi.

Chưa thay đổi bộ câu hỏi dành cho hạng xe A3/A4.

8. Kết cấu của đề tài

Luận văn được chia làm 3 phần: phần mở đầu, phần nội dung, phần kết luận.

Phần mở đầu: Trình bày lý do chọn đề tài, tình hình nghiên cứu, mục tiêu và phương pháp nghiên cứu của đề tài, cũng như những đóng góp mới và những mặt hạn chế của đề tài.

Phần nội dung:

- **Chương 1**: Giới thiệu sơ lược về ngôn ngữ Objective C như lịch sử hình thành, đặc điểm, sự khác biệt giữa Objective C và C/C++ và một số vấn đề cơ bản của Objective C.
- **Chương 2**: Lập trình ứng dụng trên iOS: giới thiệu về công cụ lập trình Xcode, thao tác tạo dự án mới, cấu trúc và các bước để tạo được một ứng dụng, cách đưa ứng dụng lên thiết bị thật và cách chia sẻ ứng dụng cho người dùng khác.
- **Chương 3**: Xây dựng ứng dụng luyện thi lý thuyết lái xe: phân tích ứng dụng, ý tưởng, các chức năng, cách lưu trữ dữ liệu, …
- **Chương 4**: Giao diện ứng dụng.

Phần kết luận: Trình bày kết luận về đề tài, kiến nghị về hướng nghiên cứu tiếp theo của đề tài.

CHƯƠNG 1: GIỚI THIỆU SƠ LƯỢC VỀ NGÔN NGỮ OBJECTIVE-C

1.1. Khái niệm và lịch sử

1.1.1. Giới thiệu về ngôn ngữ Objective C

Objective C là ngôn ngữ lập trình hướng đối tượng được xây dựng dựa trên nền tảng ngôn ngữ C. Là ngôn ngữ chính được dùng để lập trình ứng dụng trên hệ điều hành OS X dùng cho MacBook, iMac và iOS dùng cho thiết bị di động như iPhone, iPad.

1.1.2. Lịch sử hình thành và phát triển

Đầu những năm 80, Brad J.Cox và Tom Love đã thiết kế ra ngôn ngữ Objective-C dựa trên ngôn ngữ Smalltalk-80, một trong những ngôn ngữ lập trình hướng đối tượng đầu tiên.

Từ năm 1988, công ty NeXT Software nắm giữ bản quyền ngôn ngữ Objective-C đã phát triển thêm các bộ thư viện và cả môi trường phát triển cho nó là NEXTSTEP.

Năm 1994, NeXT Computer phối hợp với Sun Microsystems chuẩn hoá lại NEXTSTEP trong bản đặc tả có tên là OPENSTEP (bản hiện thực GNUStep) bao gồm cả Linux kenel và môi trường phát triển GNUStep (LinuxSTEP).

Cuối tháng 12 năm 1996, Apple mua lại công ty NeXT Software và môi trường NEXTSTEP/OPENSTEP trở thành thành phần cốt lõi của hệ điều hành OS X. Phiên bản chính thức của môi trường này có tên là Cocoa.

Năm 2007, Apple tung ra bản nâng cấp cho ngôn ngữ Objective-C và chính thức giới thiệu iPhone.

Mới đây tại WWDC 2014, Apple đã công bố kế hoạch thay thế Objective-C bằng một ngôn ngữ hoàn toàn mới là Swift và theo Apple nó sẽ giúp phát triển ứng dụng nhanh hơn, dễ dàng, trực quan hơn; được mô tả như là "Có Objective-C mà không có C".

1.2. Đặc điểm của Objective C

1.2.1. Đặc điểm cơ bản của Objective C

– Là ngôn ngữ hướng đối tượng.

– Mở rộng từ C.

– Nhẹ nhàng (không sử dụng virtual machine)

– Mềm dẻo (do mở rộng từ ngôn ngữ C nên có thể dùng C thuần cấu trúc)

– Reflection (có hỗ trợ)

– Nil dùng để thay thế cho NULL trong C, vì ta có thể gửi thông điệp cho nil nhưng đối với NULL thì không.

– Kiểu dữ liệu logic (bool/boolean) trong Objective-C chỉ có giá trị trả về là YES/NO mà không dùng giá trị true/false hoặc 1/0.

– Khái niệm methods và message được sử dụng mang ý nghĩa như nhau đối với Objective-C theo đó message có những thuộc tính đặc biệt. Moto message có thể chuyền động từ object này tới một object khác. Việc gọi thông điệp trên một object không có nghĩa là object đó sẽ thực hiện message mà nó có thể chuyển tiếp tới một object khác chưa biết trước; tóm lại khả năng đáp trả thông điệp có thể là trực tiếp hoặc gián tiếp.

Do có tính mềm dẻo nên khi sử dụng Objective-C cần chú ý việc sử dụng cú pháp C trộn lẫn với cú pháp chính thống của Objective-C là hoàn toàn có thể chấp nhận, tuy nhiên về mặt hình thức và code sẽ không được chuyên nghiệp.

1.2.2. Khác biệt căn bản giữa C/C++ với Objective C

Mặc dù được mở rộng từ ngôn ngữ C nhưng Objective-C có cấu trúc hoàn toàn khác so với C. Bảng dưới đây tóm tắt một số khác biệt cơ bản về các từ khoá thường dùng giữa C và Objective-C.

Bảng 1.1: Một số khác biệt cơ bản về từ khoá

C/C++	Objective-C
#include “library.h”	#import “library.h”
this	self
private: , protected: , public:	@private, @protected, @lublic
Y = new MyClass()	Y = [[MyClass alloc] init]

try, throw, catch, finally	@try, @throw, @catch, @finally

Ngoài ra, cách gọi hàm trong Objective-C khá giống với C tuy nhiên cách viết lại hoàn toàn khác khiến chúng ta khó có thể hiểu khi chỉ mới nhìn sơ qua.

Bảng 1.2: Khác biệt về cách gọi hàm

	C/C++	Objective-C
Gọi method	myObject.SomeMethod();	[myObject SomeMethod]
Trường hợp có 1 parameter	myObject.SomeMethod(20);	[myObject SomeMethod:20]
Trường hợp có nhiều parameter	myObject.SomeMethod(20, 40);	[myObject SomeMethod: 20: 40]
Gọi hàm theo nhiều lớp	myObject.SomeMethod().OtherMethod();	[[myObject SomeMethod] OtherMethod]

1.3. Cơ bản về Objective C

1.3.1. Các kiểu dữ liệu và toán tử cơ bản trong Objective C

1.3.1.1. Biến và cách khai báo biến

Biến là nơi lưu trữ dữ liệu và có thể thay đổi khi chương trình thực thi.

Quy tắc đặt tên biến:

- Ngôn ngữ Objective-C có phân biệt kí tự hoa thường.
- Không được có dấu tiếng việt.
- Không có khoảng trắng.
- Không được bắt đầu bằng số.
- Không được chứa các kí tự đặc biệt (trừ dấu gạch dưới "_").
- Là duy nhất và không được trùng tên với các từ khoá có sẵn của Objective-C.

Cú pháp:

<Kiểu dữ liệu> <Tên biến>;

hoặc

<Kiểu dữ liệu> <Tên biến> = <giá trị>;

1.3.1.2. Kiểu dữ liệu

Kiểu dữ liệu là một sự quy định về cấu trúc, miền giá trị của dữ liệu và tập các phép toán tác động lên miền giá trị đó. Nó giúp cho trình biên dịch xác định được loại dữ liệu mà muốn lưu trữ từ đó sẽ cấp phát bộ nhớ tương ứng với loại dữ liệu mà chúng ta sử dụng.

Tương tự các ngôn ngữ lập trình khác, Objective-C cũng có các kiểu dữ liệu cơ bản nhất và thường được sử dụng nhất được thống kê trong bảng sau:

Bảng 1.3: Các kiểu dữ liệu

Loại	Tên Kiểu	Số ô nhớ
Kí tự	Char	1 byte
Số nguyên	Short	2 byte
	Int	4 byte
	Long	
Số thực	Float	4 byte
	Double	8 byte
Logic	bool	1 byte

Ngoài ra còn có các kiểu dữ liệu khác như: unsigned char, unsigned int, unsigned short, long long, unsigned long, unsigned long long, …

1.3.1.3. Toán tử

Để thao tác với dữ liệu ta sử dụng các toán tử, đó là các kí hiệu đặc biệt dùng để thực hiện các phép toán số học cơ bản.

Bảng 1.4: Các toán tử

Loại toán tử	Tên phép toán	Kí hiệu	Ví dụ
Toán tử gán		=	A = B
Toán tử số học	Cộng	+	A + B
	Trừ	-	A – B

	Nhân	*	A * B
	Chia	/	A / B
	Lấy phần dư	%	A % B
Toán tử quan hệ	Bằng	==	A == B
	Khác	!=	A != B
	Lớn hơn	>	A > B
	Lớn hơn bằng	>=	A >= B
	Nhỏ hơn	<	A < B
	Nhỏ hơn bằng	<=	A <= B
Toán tử logic	Not	!	!A
	And	&&	A && B
	Or	||	A || B
Toán tử điều kiện	Nếu … thì	… ? … : …	ĐK ? A : B

1.3.2. Phương thức

Phương thức (Function) là một loạt các câu lệnh dùng để thực hiện một số công việc định sẵn mà các công việc này được sử dụng nhiều lần trong chương trình.

Có 2 loại phương thức: phương thức không có tham số truyền vào và phương thức có tham số truyền vào.

1.3.2.1. Phương thức không có tham số

Cú pháp:

-(<Kiểu dữ liệu>) <Tên hàm>

{

//Các câu lệnh

[return <Biểu thức>];

}

Trong đó:

– Dấu "-" là bắt buộc để trình biên dịch nhận biết đây là hàm thực thi.

– Kiểu dữ liệu là kiểu dữ liệu của biểu thức được trả về sau khi hàm thực thi.

– Tên hàm: đặt theo quy tắc tương tự như quy tắc đặt tên biến.

1.3.2.2. Phương thức có tham số

Cú pháp:

-(<Kiểu dữ liệu>) <Tên hàm> : (<Kiểu dữ liệu>) <tham số 1> [Mô tả] : (<Kiểu dữ liệu>) <tham số 2> [Mô tả]

{

//Các câu lệnh

[return <Biểu thức>];

}

Trong đó:

– Dấu "-" là bắt buộc để trình biên dịch nhận biết đây là hàm thực thi.

– Kiểu dữ liệu là kiểu dữ liệu của biểu thức được trả về sau khi hàm thực thi.

– Tên hàm đặt theo quy tắc tương tự như quy tắc đặt tên biến.

– Dấu ":" là bắt buộc để ngăn cách giữa các tham số.

– Mô tả: có thể có hoặc không, dùng để mô tả chi tiết cho tham số.

1.3.2.3. Cách gọi phương thức

– Đối với phương thức không có tham số

[self <tên hàm>];

– Đối với phương thức có tham số

[self <tên hàm>: <tham số 1> : <tham số 2>];

Trong đó từ khoá self là con trỏ đại diện cho lớp hiện tại, tương đương với từ khoá this trong C.

1.4. Ưu, nhược điểm của Objective C

1.4.1. Ưu điểm

– Là ngôn ngữ chính được sử dụng để phát triển các ứng dụng cho các sản phẩm Apple.

– Là ngôn ngữ lập trình được phát triển đặc biệt để làm việc với template CoCoa Touch, điều này có nghĩa rằng nó có quyền truy cập vào các thư viện phát triển của Apple.

– Objective C runtime: C/C++ được "biên dịch" theo ngôn ngữ lập trình, có nghĩa là mã nguồn viết bởi các lập trình viên được dịch sang ngôn ngữ máy tạo ra một tập tin thực thi chạy trên hệ điều hành của người dùng. Ngôn ngữ biên dịch bị giới hạn trong cách làm việc của các lập trình viên như: các quyết định liên quan đến việc phân bổ bộ nhớ và tạo đối tượng xảy ra trong thời gian biên dịch mà không phải trong thời gian thực thi chương trình (runtime). Trong khí đó, một chương trình được viết bằng Objective C sẽ năng động hơn do nó có khả năng thu thập thông tin về chương trình để đưa ra quyết định liên quan đến bộ nhớ hoặc loại dữ liệu thay vì phải quyết định thực hiện trong thời gian mã hoá.

1.4.2. Nhược điểm

– Namespace: Objective C không chứa namespace. Trong các ngôn ngữ khác, đặc biệt là C++, namespace có chứa các chức năng được xác định bởi một tên. Chức năng trong một namespace chỉ tồn tại trong phạm vi namespace đó; có nghĩa là một chức năng có tên giống hệt nhau trong namespace khác có thể được sử dụng bới các lập trình viên mà không cần bất kỳ việc thay đổi tên nào. Điều này cho phép các lập trình viên có thể thêm một loạt các thư viện mà không lo về việc tên hàm bị trùng hay mâu thuẫn nhau. Mặt khác, trong Objective C, khi thêm hai thư viện với chức năng trùng tên có thể gây ra lỗi nếu lập trình viên không cẩn thận bởi vì trình biên dịch không có cách nào để biết chức năng đó thực sự là gì để có thể gọi đến.

– Khả năng di chuyển: Objective C phụ thuộc rất nhiều vào các template CoCoa cho các chức năng; trong khi đây là một lợi ích cho những nhà phát triển của Apple thì các nhà phát triển Windows sẽ không thể tìm thấy những thành công tương tự khi viết các ứng dụng Windows trong Objective C. Tuy nhiên, Apple cũng không cần phải lo lắng về vấn đề này trừ khi họ đang tìm kiếm một nền tảng di động mới.

1.5. Tiểu kết

Chương này cung cấp sơ lược về ngôn ngữ Objective C, tìm hiểu được lịch sử hình thành, các đặc điểm, một số khác biệt so với ngôn ngữ C/C++ và nhưng ưu nhược điểm của Objective C so với C/C++. Bên cạnh đó còn giới thiệu cơ bản về ngôn ngữ Objective C như: biến, kiểu dữ liệu, phương thức, …

CHƯƠNG 2: LẬP TRÌNH ỨNG DỤNG TRÊN iOS

2.1. Môi trường lập trình

2.1.1. Công cụ

Để lập trình ứng dụng trên iOS trước tiên cần có bộ công cụ phát triển do chính Apple cung cấp miễn phí trên AppStore gọi là Xcode. Vì bộ công cụ phát triển này chạy trên nền hệ điều hành Mac OS X nên muốn cài đặt và sử dụng nó cần phải có một chiếc máy chạy hệ điều hành Mac OS X.

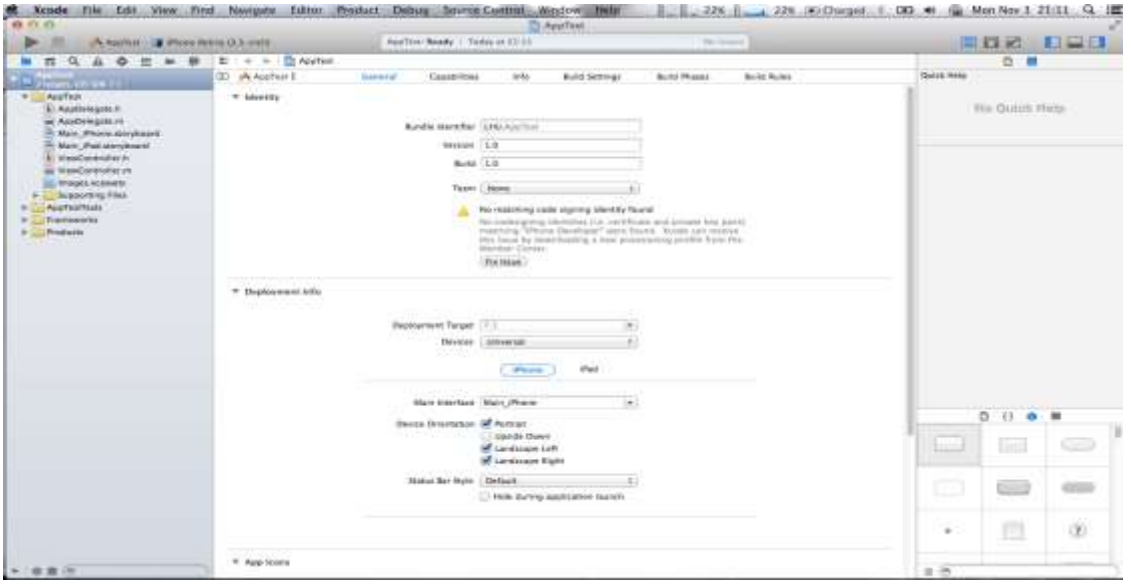
Trước đây, để có thể lập trình ứng dụng trên iOS bắt buộc cần phải có một máy Mac (iMac, MacBook,…); nhưng những năm trở lại đây việc lập trình ứng dụng cho các thiết bị của Apple đã trở nên đơn giản hơn rất nhiều do không nhất thiết phải sử dụng máy Mac mà có thể lập trình trên các máy không phải là máy Mac (máy chạy môi trường Windown) bằng cách cài hệ điều hành OS X trên máy ảo hoặc cài đặt bản hackintosh trực tiếp trên máy tính.

Cuối cùng, tải về và cài đặt phần mềm Xcode từ AppStore là có thể bắt đầu quá trình lập trình iOS.

2.1.2. Giới thiệu sơ lược về công cụ Xcode

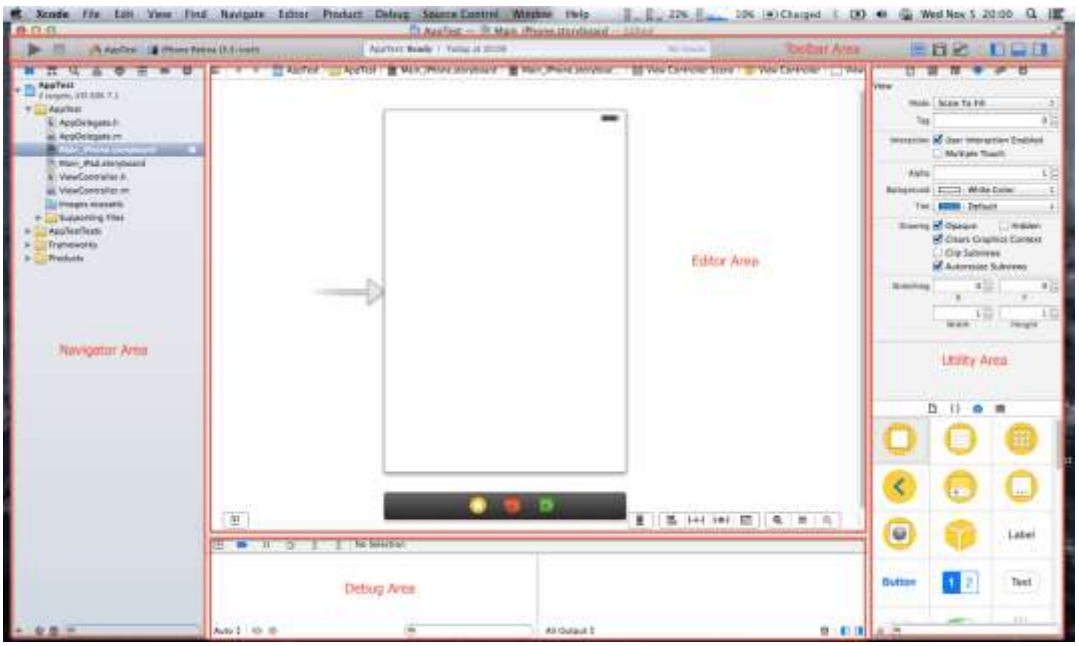
2.1.2.1. Xcode là gì?

Xcode là bộ công cụ phát triển ứng dụng được Apple cung cấp cho các lập trình viên để lập trình ứng dụng cho các thiết bị chạy hệ điều hành của Apple (OS X, iOS). Phiên bản mới nhất hiện nay trên trang developer của Apple là phiên bản Xcode 6 BETA; trên AppStore là phiên bản Xcode 5.[11]



Hình 2.1: Phần mềm Xcode (phiên bản 5.1.1)

2.1.2.2. Giao diện Xcode



Hình 2.2: Giao diện làm việc của Xcode

Giao diện làm việc của Xcode gồm 5 phần chính:

– Debug area: vùng hỗ trợ trong quá trình debug lỗi của chương trình.

– Toolbar area: vùng chứa các công cụ tiện ích giúp đơn giản trong việc debug ứng dụng, chạy chương trình, lựa chọn máy ảo (iOS Simulator),…

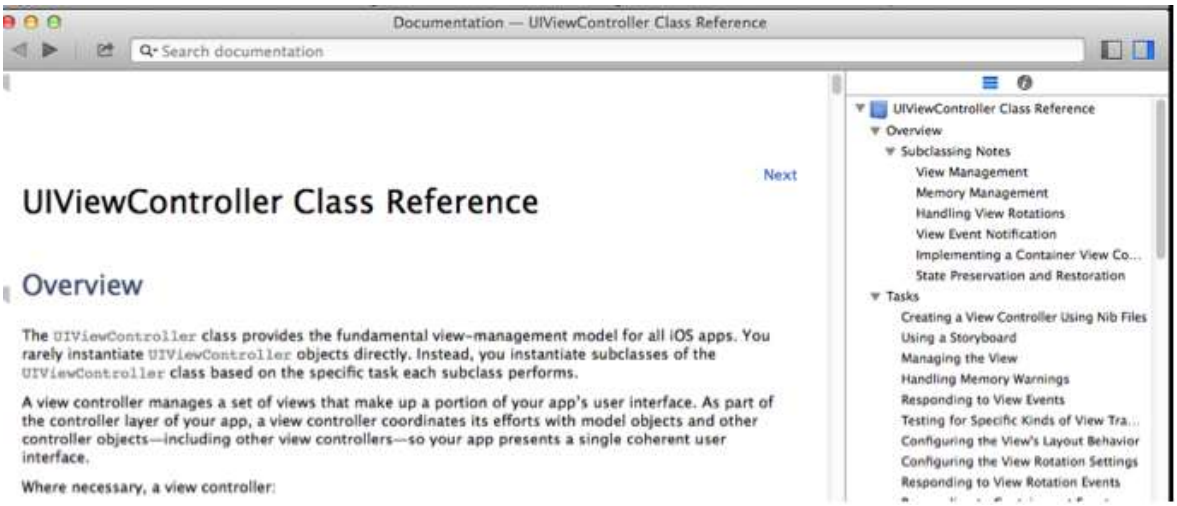
– Editor area: vùng thiết kế giao diện, viết và chỉnh sửa code.

– Utility area: vùng cho phép tuỳ chỉnh các tham số, giá trị thuộc tính của các đối tượng trên giao diện cũng như cho phép kéo thả và sử dụng các đối tượng có sẵn để thiết kế giao diện như: button, label, … hay các đoạn code mẫu được cung cấp sẵn như: if, switch, for, …

– Navigator area: cung cấp một cách nhìn trực quan, tổng thể và tiện lợi cho việc quản lý ứng dụng, xem thông báo lỗi, tìm kiếm một đoạn code trong chương trình hay kiểm tra mức độ hoạt động của RAM, CPU khi đang chạy ứng dụng, …

Xcode cũng cung cấp chế độ gỡ lỗi thông minh, hỗ trợ trong việc phát hiện lỗi, cảnh báo lỗi và gợi ý cách khắc phục. Ngoài ra, kèm theo Xcode là bộ tài liệu hướng dẫn chi tiết từng bước tiện lợi nhằm hỗ trợ người dùng trong việc lập trình. [1]

Hình 2.3: Chế độ gỡ lỗi thông minh



Hình 2.4: Bộ tài liệu hướng dẫn đi kèm

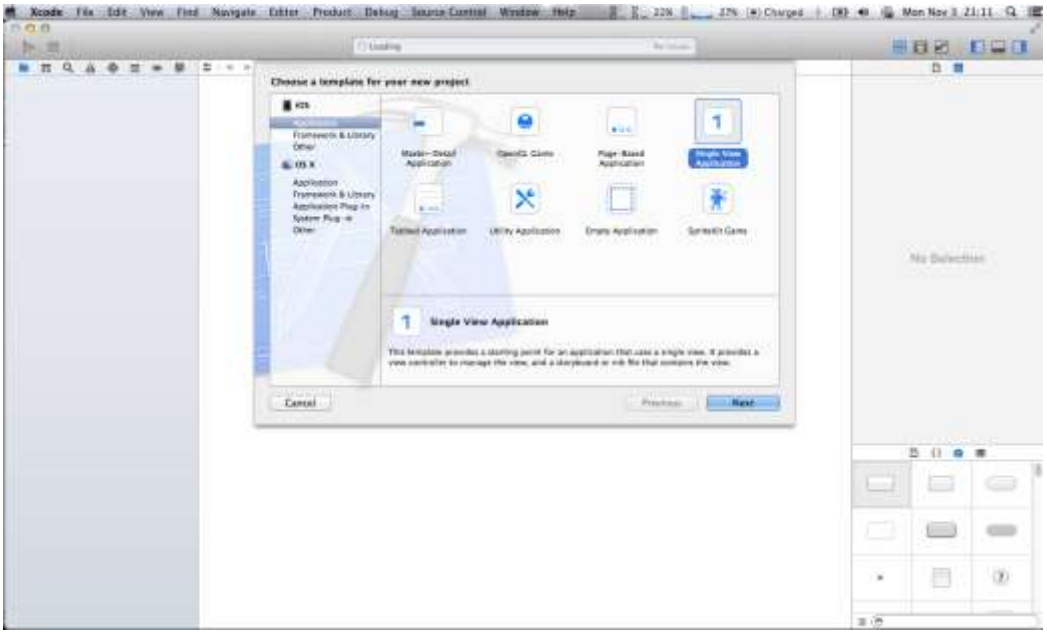
2.1.3. Thao tác tạo một dự án mới

Khi khởi động Xcode, cửa sổ hiện ra cho phép ta tạo một project mới hoặc mở một project mới được mở gần đây; để tạo một project mới ta chọn Create a new

Xcode project. Một cách khác để tạo project mới là trên menu chọn File -> New -> Project… hoặc bấm tổ hợp phím tắt fn + command(⌘) + N.

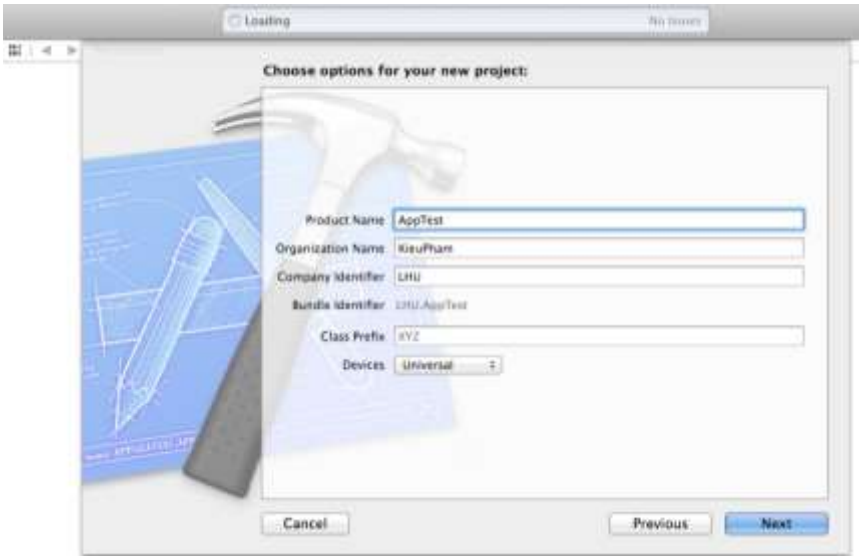
Hình 2.5: Tạo project mới

Sau khi chọn new project, Xcode sẽ hiển thị một cửa sổ yêu cầu lựa chọn một hình thức lập trình cho project được tạo (ứng dụng cho iPhone, iPad hay Mac OS X, …) và loại template cho ứng dụng (Single View, Empty View hay OpenGL Game, …). Đơn giản nhất ta nên chọn Single View; sau đó click Next.



Hình 2.6: Lựa chọn hình thức lập trình và template

Tiếp theo, ta điền thêm các thuộc tính cho project như tên ứng dụng (Product Name), Organization Name, Company Identifer. Sau đó tiến hành lựa chọn loại thiết bị mà ứng dụng sẽ chạy trên nó (iPhone, iPad hay Universal để viết ứng dụng chạy trên cả 2 loại thiết bị trên). Cuối cùng chọn nơi lưu trữ project và click vào nút Create.



Hình 2.7: Điền các thông tin và thuộc tính cho project

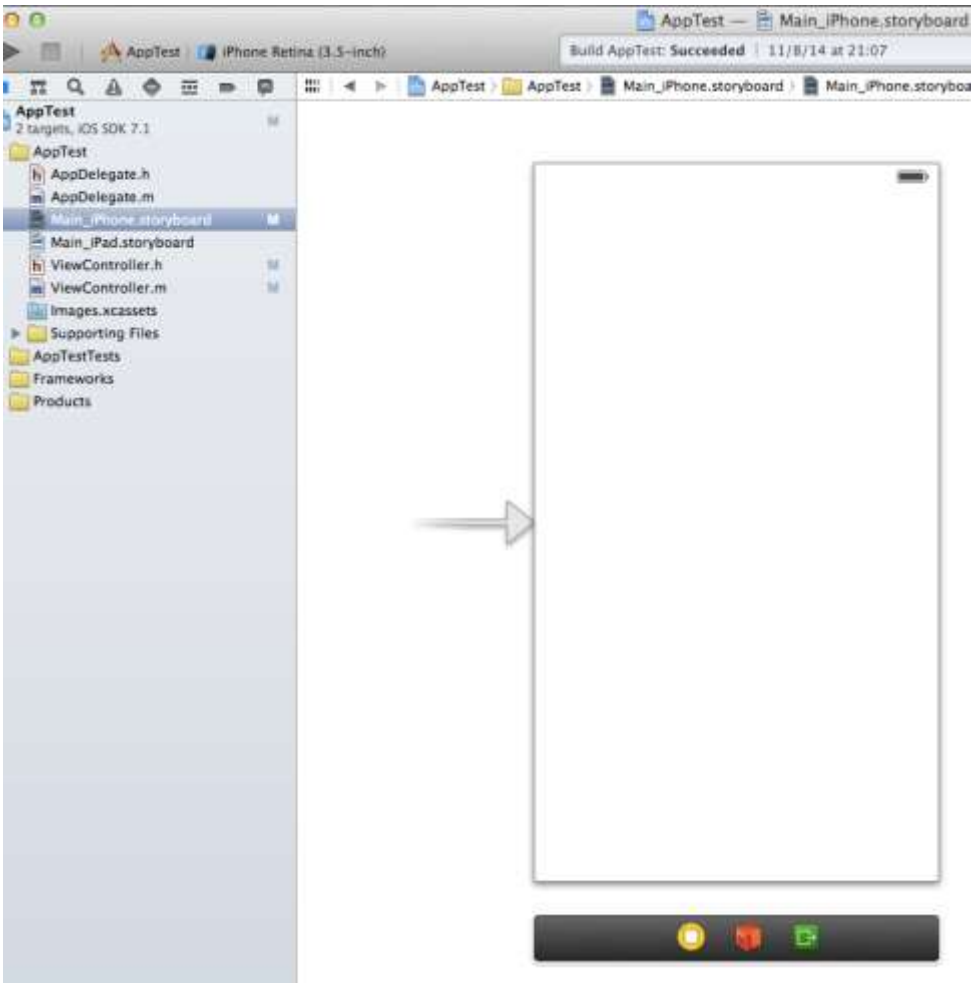
Với các bước trên, ta đã tạo được một project để chuẩn bị cho việc lập trình ứng dụng bằng công cụ Xcode.

2.2. Cấu trúc và các bước để tạo một ứng dụng iOS

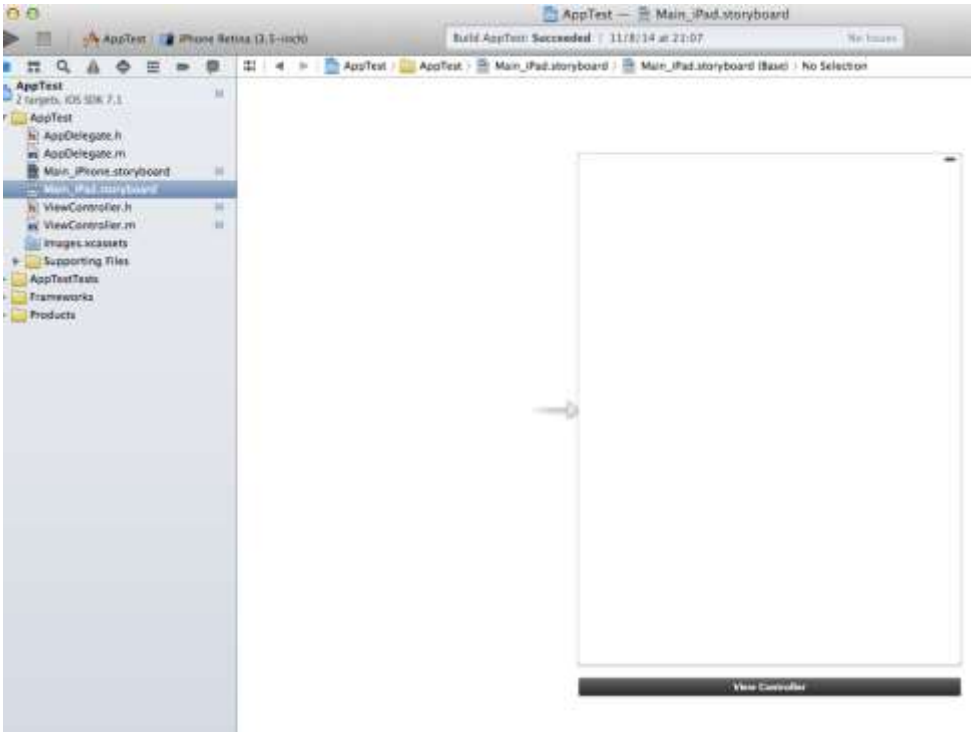
2.2.1. Cấu trúc một ứng dụng iOS

Cấu trúc một ứng dụng trên iOS gồm có tối thiểu 3 file quan trọng:

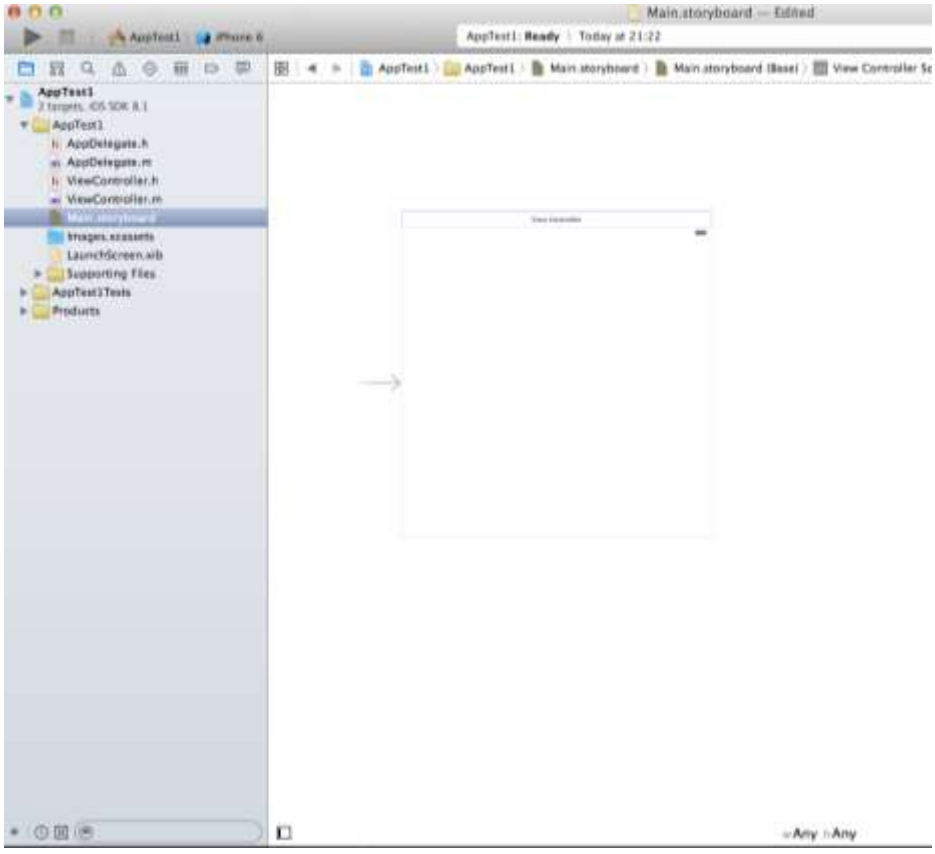
– File Main.storyboard: là file giao diện dùng để thiết kế giao diện hiển thị cho ứng dụng. Tại Xcode 5, nếu chọn thiết kế cho cả iPad và iPhone thì sẽ có 2 file giao diện là Main_iPhone.storyboard dùng để thiết kế giao diện cho iPhone và Main_iPad.storyboard dùng để thiết kế giao diện cho iPad. Ngược lại, với Xcode 6 chỉ xuất hiện 1 file Main.storyboard dùng để thiết kế cho cả 2 loại thiết bị.



Hình 2.8: File Main_iPhone.storyboard (Xcode 5 trở xuống)



Hình 2.9: File Main_iPad.storyboard (Xcode 5 trở xuống)



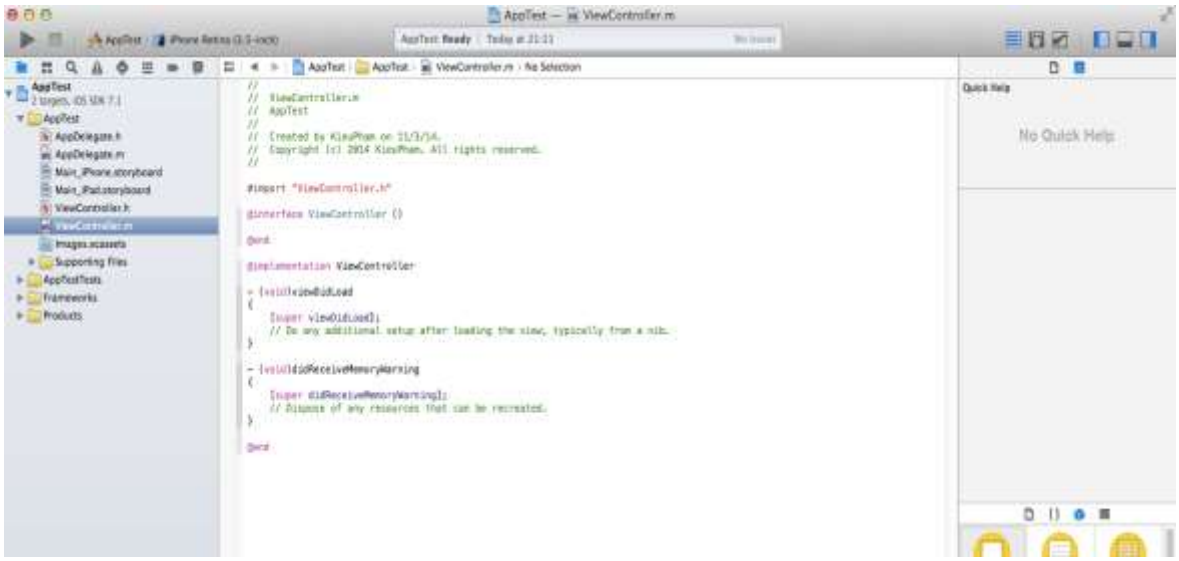
Hình 2.10: File Main.storyboard (Xcode 6)

➢ Lưu ý: Vì Xcode 6 chỉ hỗ trợ 1 file thiết kế dành cho cả 2 loại thiết bị nên tất cả các đối tượng sau khi thiết kế đều được AutoSize cho phù hợp với thiết bị. Mặc dù từ Xcode 5 trở xuống vẫn hỗ trợ 2 file thiết kế dành riêng cho iPhone và Ipad; tuy nhiên các thiết bị thật thường có sự khác biệt nhau về độ phân giải cho nên trong quá trình thiết kế các đối tượng cũng phải được cấu hình AutoSize.

– File .h: viết tắt của từ Header; dùng để kết nối và khai báo các đối tượng trong file .storyboard trước khi muốn dùng các đối tượng này để lập trình cho người dùng tương tác.

Hình 2.11: File .h

– File .m: viết tắt của từ Main; dùng để triển khai chi tiết các sự kiện đã được khai báo trong file .h.



Hình 2.12: File .m

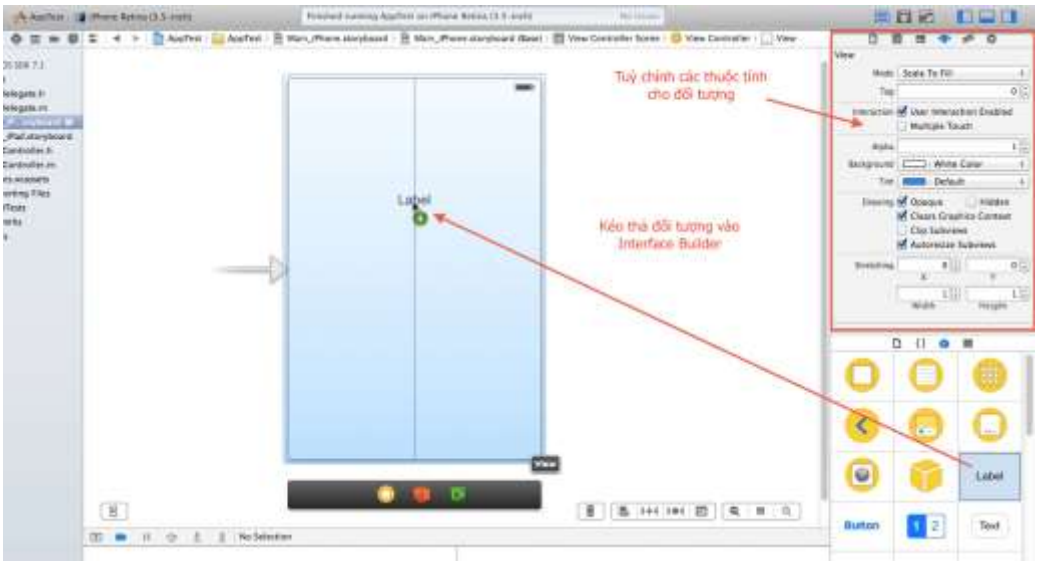
2.2.2. Các bước tạo một ứng dụng

2.2.2.1. Thiết kế giao diện

Giao diện ứng dụng trong Xcode được thiết kế thông qua Interface Builder, các đối tượng được cung cấp trong Utility area.

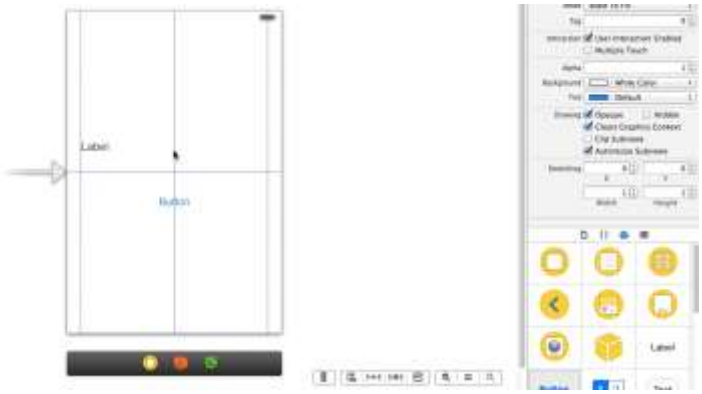
Để thiết kế giao diện, ta mở file Main.storyboard và kéo thả đối tượng cần dùng từ vùng Library trong Utility area vào Interface Builder. Tại đây có thể điều chỉnh, di chuyển, sắp xếp vị trí của các đối tượng theo ý tưởng, bản thiết kế.

Sau khi kéo thả đối tượng vào Interface Builder, ta có thể điều chỉnh các thuộc tính cho đối tượng như màu sắc, font chữ, màu chữ, … trong vùng Inspector pane thuộc Utility area.



Hình 2.13: Thiết kế giao diện

Xcode còn có tính năng hỗ trợ canh chỉnh thông qua các đường kẻ đứt nét màu xanh giúp dễ dàng xác định vị trí đặt các đối tượng sao cho cân đối và không bị lệnh khi Run ứng dụng trên thiết bị.

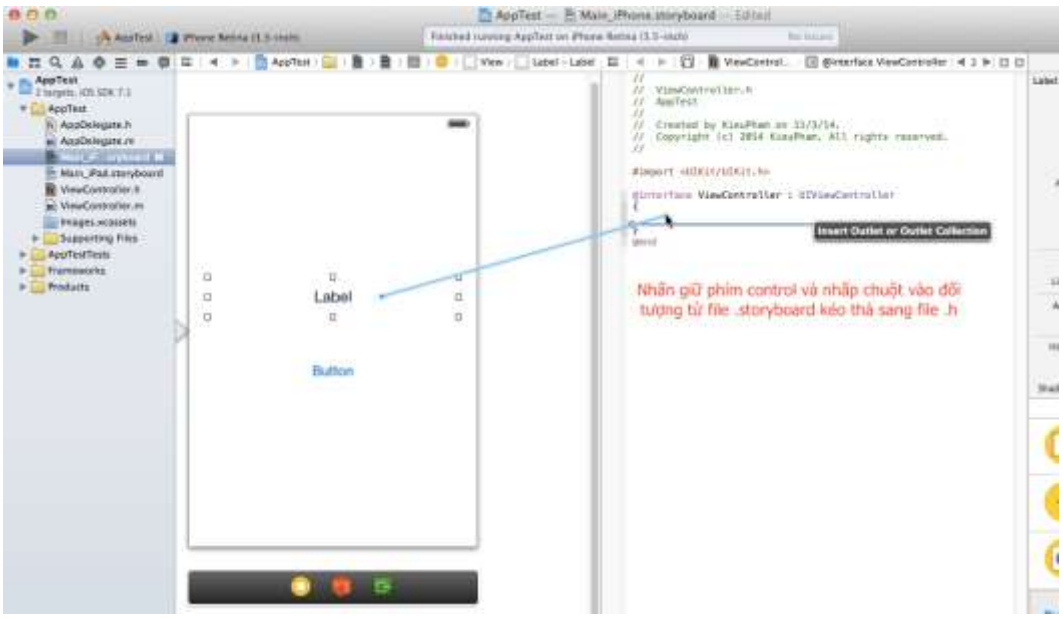


Hình 2.14: Hỗ trợ canh chỉnh

2.2.2.2. Khai báo và kết nối đối tượng

Việc khai báo, kết nối đối tượng trong Xcode được gọi là ánh xạ và được thực hiện trong file .h.

Xcode hỗ trợ việc khai báo, kết nối đối tượng một cách đơn giản, nhanh chóng đó là nhấn kết hợp phím Ctrl và nhấp chuột vào đối tượng trong file Main.storyboard sau đó kéo thả vào file .h. Khi đó Xcode sẽ hiển thị một hộp thoại để lựa chọn kiểu kết nối là Outlet (dùng để hiển thị thông tin hoặc cho người dùng nhập liệu) hay Action (dùng để khi người dùng tương tác với đối tượng thì sẽ thực hiện các kết quả tuỳ theo chức năng). Tiếp theo là đặt tên cho đối tượng; sau đó click Connect.



Hình 2.15: Khai báo và kết nối đối tượng

Ngoài ra, ta còn có thể viết code khai báo trực tiếp trong file .h sau đó kết nối tới file Main.storyboard.

2.2.2.3. Viết code xử lý sự kiện cho đối tượng

Code xử lý các sự kiện, đối tượng được viết trong file .m.

Trong file này đã có sẵn hai sự kiện được phát sinh tự động đó là viewDidLoad (hàm chạy đầu tiên khi ứng dụng được mở lên) và didReciveMemoryWarning (hàm dùng để giải phóng bộ nhớ); 2 sự kiện này có thể giữ nguyên, thêm code xử lý hoặc xoá bỏ (không khuyến khích xoá bỏ vì có thể dẫn đến lỗi trong quá trình build ứng dụng).

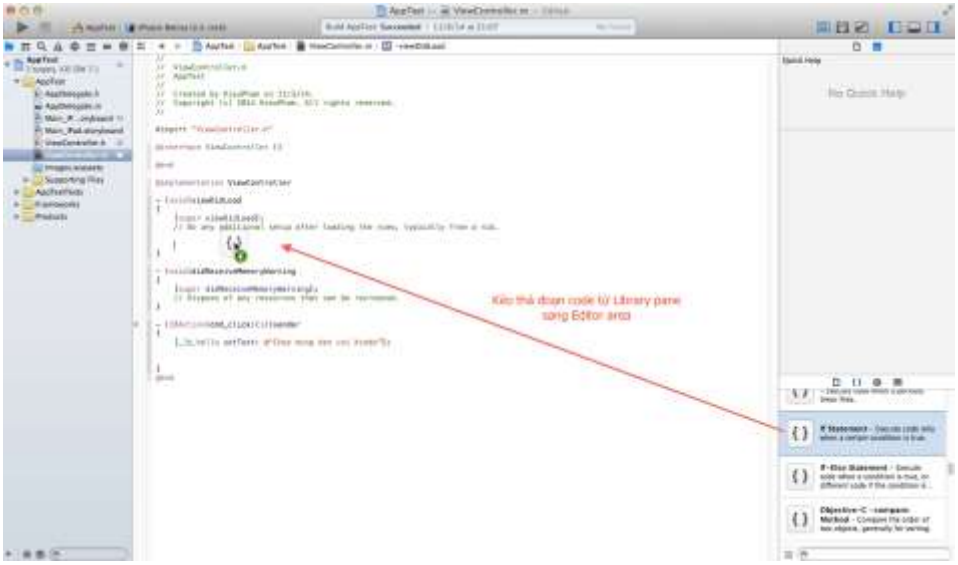
Hình 2.16: Hai sự kiện phát sinh tự động

Xcode hỗ trợ phát hiện lỗi trong quá trình viết code và thông báo bằng hình tròn màu đỏ ngay tại câu lệnh sai. Ngoài ra, Xcode còn có chế độ cảnh báo với biểu tượng tam giác màu vàng tại đoạn code mà công cụ cho rằng cần phải cải thiện; không chỉ thế Xcode còn cung cấp một chế độ gợi ý sửa lỗi trong quá trình viết code. Tuy nhiên, những hỗ trợ trên chỉ khắc phục một số lỗi cơ bản về cú pháp do vậy các lỗi logic xảy ra trong quá trình viết code hay debug ứng dụng chúng ta phải tự giải quyết.

Hình 2.17: Chế độ phát hiện và gợi ý sửa lỗi

Hình 2.18: Chế độ cảnh báo

Tiện lợi hơn nữa là Xcode còn hỗ trợ sẵn một số đoạn code mẫu trong thư viện (if, switch, …) được đặt trong vùng Library pane của Utility area; khi cần sử dụng đoạn code nào chỉ cần kéo thả đoạn code đó từ Library pane sang Editor area. Nếu có một đoạn code mà ta muốn lưu nó lại để sử dụng sau này mà không mất công phải viết lại thì có thể chọn đoạn code đó và kéo nó vào vùng Library pane sau đó đặt tên cho đoạn code và lưu lại.



Hình 2.19: Code mẫu được hỗ trợ

2.3. Cách đưa ứng dụng lên thiết bị iPhone, iPad

2.3.1. Build ứng dụng trực tiếp từ xcode lên thiết bị

Mặc dù Apple đã cung cấp công cụ giả lập có tên gọi là iOS Simulator được đính kèm theo Xcode, tuy nhiên nó vẫn còn một số hạn chế nhất định mà chỉ có thể có trên thiết bị thật. Chính vì thế việc build ứng dụng trên thiết bị thật để có thể test một cách chính xác nhất là một nhu cầu cần thiết đối với những lập trình viên.

Có 2 cách để có thể build ứng dụng trên thiết bị thật.

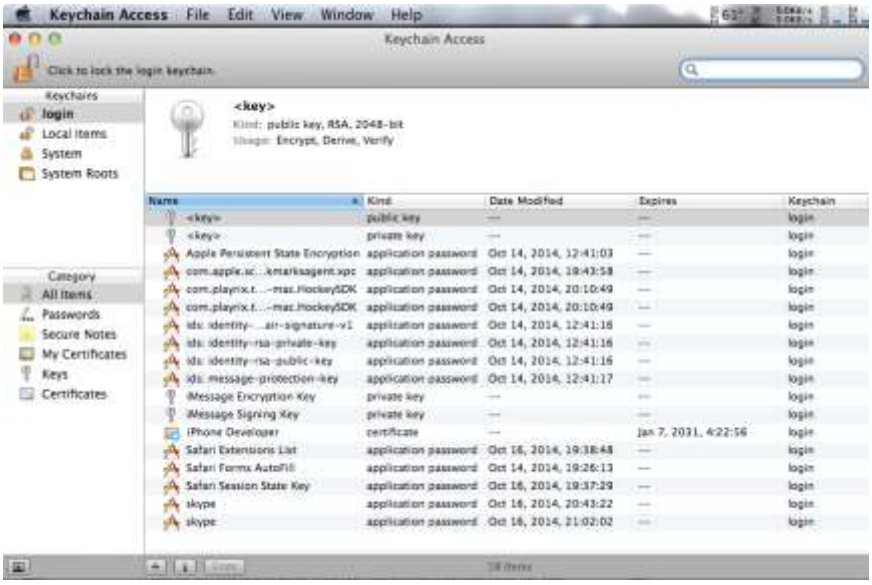
2.3.1.1. Sử dụng Apple Developer ID

Khác với Android, Apple không cho phép debug ứng dụng trực tiếp trên thiết bị thật mà cần phải có tài khoản Apple Developer ID.

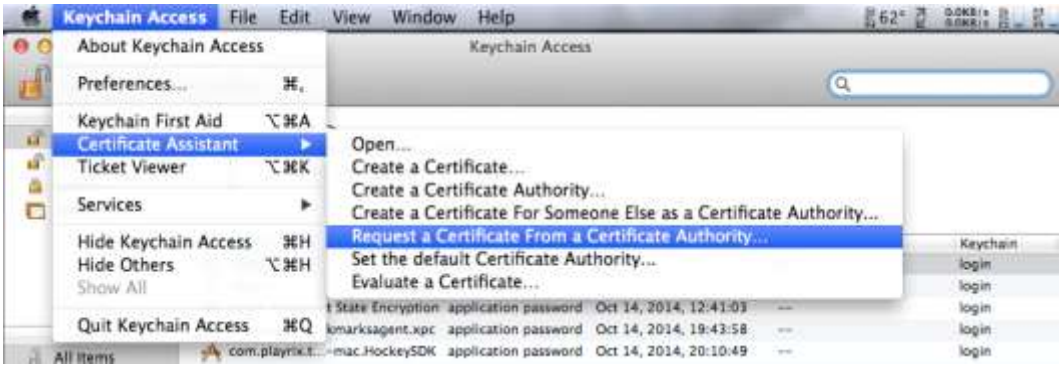
Apple Developer ID là tài khoản dành riêng cho các lập trình viên của Apple; gồm 3 loại: iOS Developer Program dành cho các lập trình viên với giá 99$/năm, iOS Developer Enterprise Program dành cho các doanh nghiệp với giá 299$/năm và iOS Developer University Program được miễn phí dành cho các trường học. Tuy nhiên đối với tài khoản iOS University Program thì chỉ có thể build được ứng dụng trên thiết bị thật mà không thể upload ứng dụng lên AppStore để chia sẻ.

Các bước build ứng dụng:

– Mở tiện ích Keychain Access trong thư mục Utilities, trên thanh menu chọn Keychain Access -> Certificate Assistant -> Request a Certificate From a Certificate Authority….

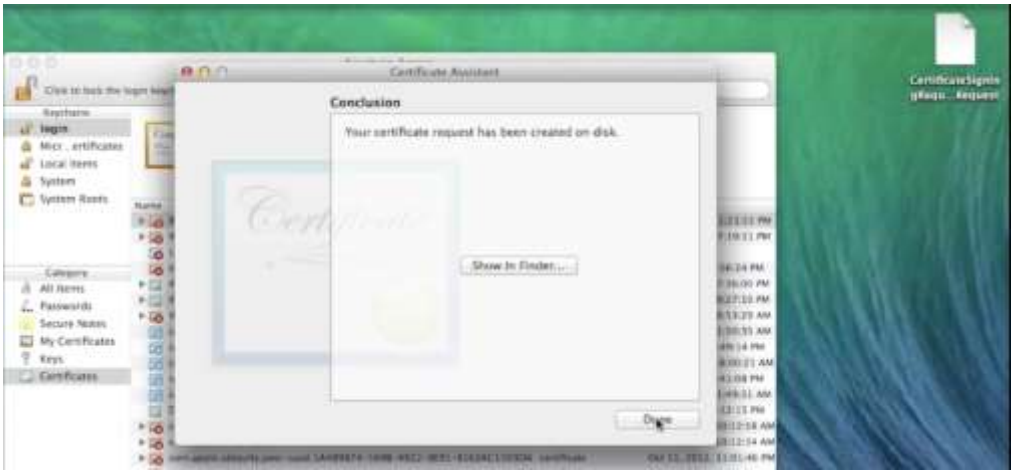


Hình 2.20: Giao diện Keychain Access



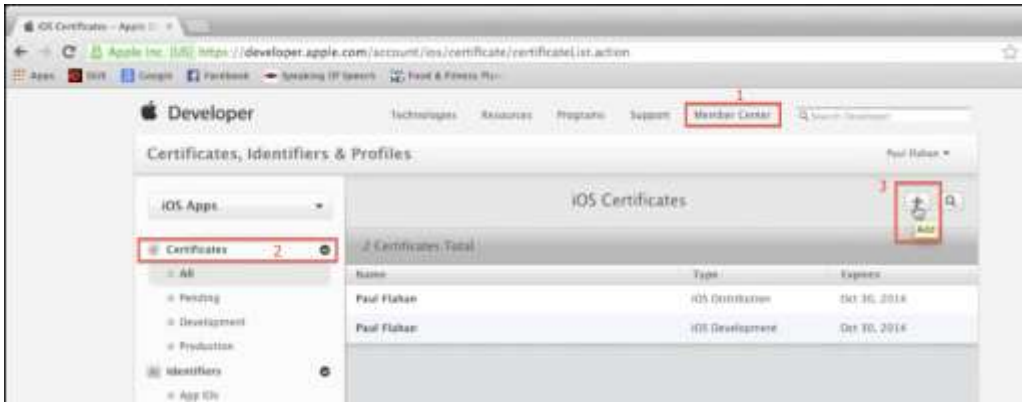
Hình 2.21: Tạo Cerrtificate

– Tại cửa sổ Certificate Assistant nhập email và tên đã đăng kí tài khoản với Apple vào ô User Email Address và Common Name. Mục Request is chọn Saved to disk -> Continues -> chọn nơi lưu file và click Save. Sau khi lưu xong trong Keychain Access sẽ hiển thị keychain mới được tạo.



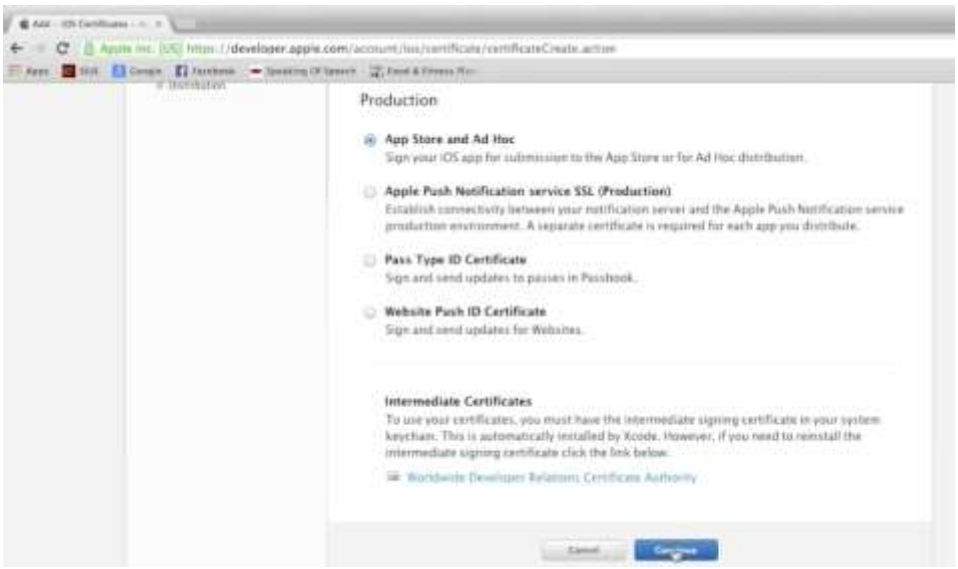
Hình 2.22: Keychain sau khi tạo xong

– Đăng nhập tài khoản Apple Developer ID trên trang chủ http://developer.apple.com. Chọn thẻ Member Center -> Certificates, Identifiers & Profiles -> Certificates -> click vào nút hình dấu cộng để tạo chứng chỉ mới.



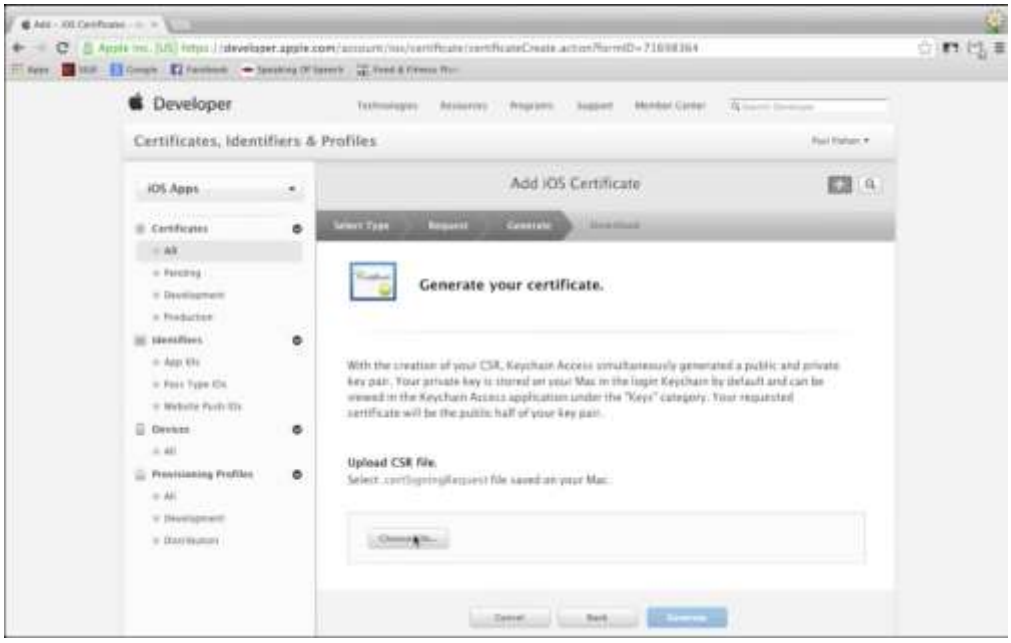
Hình 2.23: Tạo Cerrtificate trên trang Apple

– Tại phần What type of certificate do you need? chọn AppStore and AD Hoc trong mục Production -> Continue -> Continue.



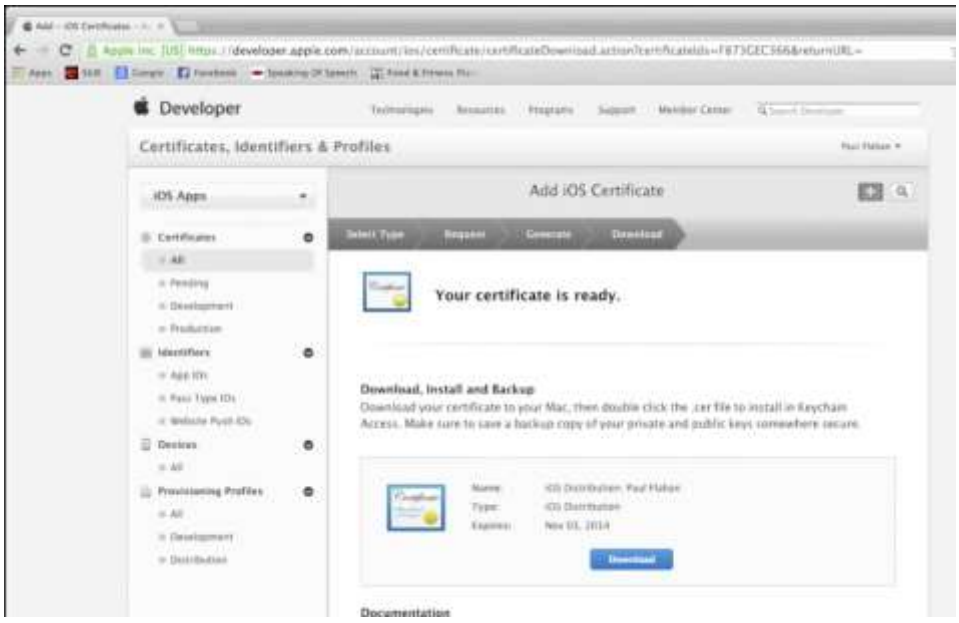
Hình 2.24: Chọn AppStore and AD Hoc

– Tại phần Generate your certificate, click Choosse File… rồi chọn đến file keychain vừa tạo ở trên -> cilck Continues để upload file keychain và tạo certificate.



Hình 2.25: Chọn đường dẫn tới file Cerrtificate vừa tạo bằng Keychain

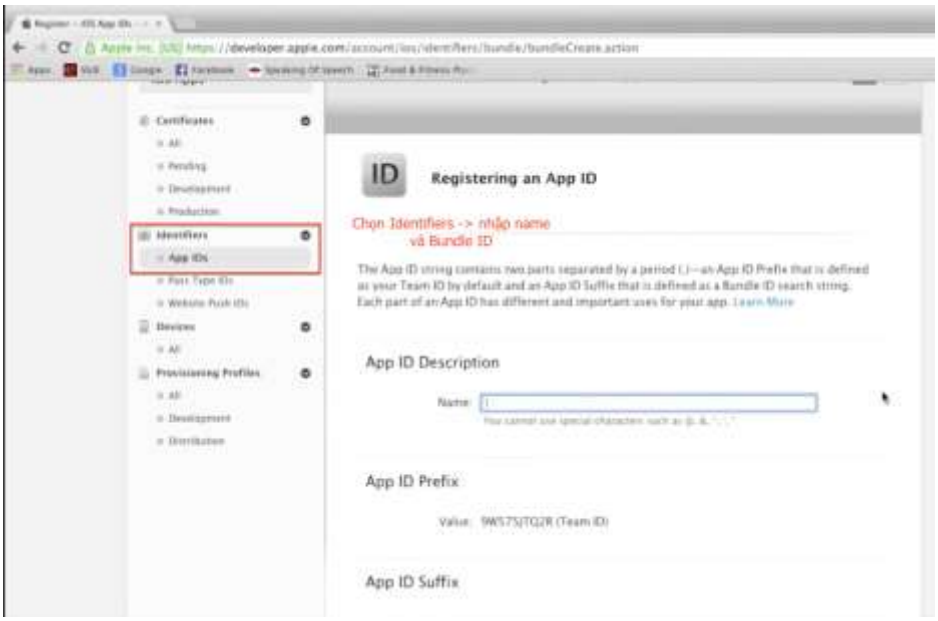
– Sau khi tạo xong click Download để download certificate về máy sau đó kéo thả vào mục Login trong Keychain Access.



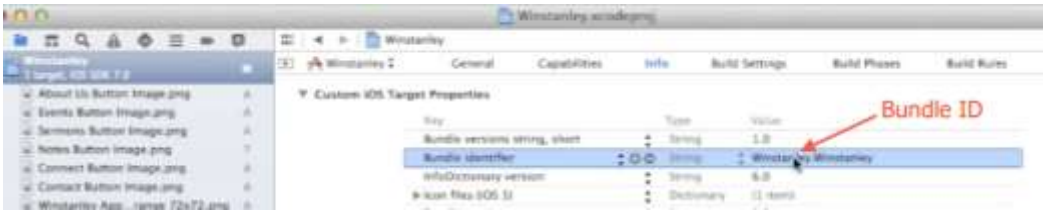
Hình 2.26: Download file Certificate

– Tạo ID App (ID App là chuỗi các kí tự dùng để nhận diện ứng dụng trên thiết bị), tại mục Identifiers sau đó nhập tên và Bundle ID. Bundle ID được lấy tại mục Bundle identifiers trong tab Info của file .xcodeproj ứng dụng -> click

Continues để bắt đầu đăng ký App IDs -> click Submit để xác nhận lại thông tin đăng ký -> click Done để hoàn tất quá trình đăng lý App IDs.

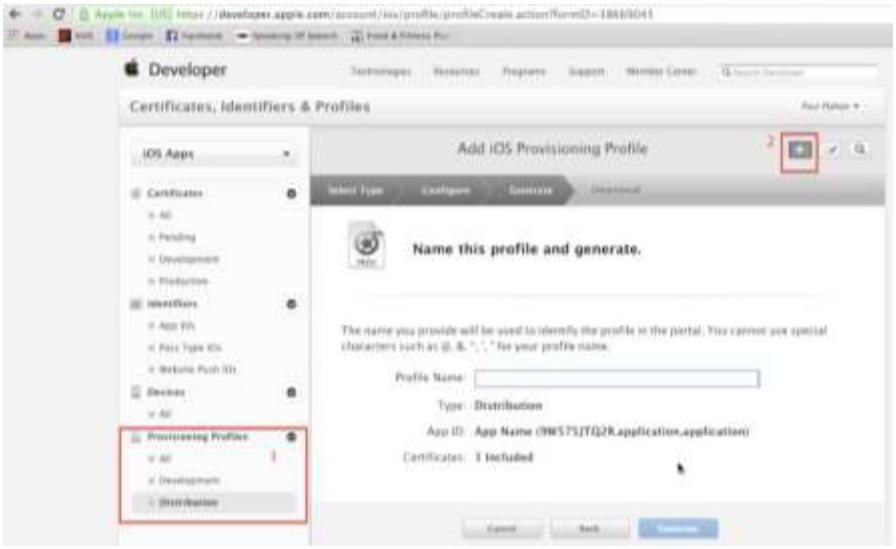


Hình 2.27: Tạo ID App



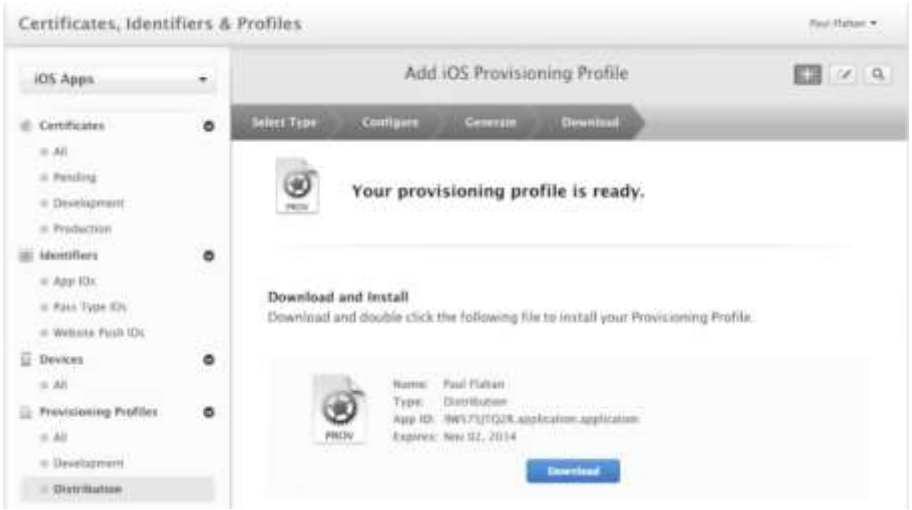
Hình 2.28: Bundle ID

– Tại mục Provisioning Profiles chọn Distribution -> click nút hình dấu cộng để tạo Provisioning Profiles mới.



Hình 2.29: Tạo Provisioning Profiles

– Phần What type of provisioning profile do you need? mục Distibution chọn Ad Hoc -> Continues. Tiếp đến chọn ID App -> Continues -> chọn Certificate -> Download về sau đó mở lên sẽ hiện thông báo Add to Library -> chọn Add to Library.



Hình 2.30: Download Provisioning Profiles

Hình 2.31: Thông báo Add to Library

Như vậy là xong phần đăng kí chứng chỉ, ta có thể mở ứng dụng bằng Xcode, chọn thiết bị và build ứng dụng.

2.3.1.2. Dùng phần mềm của hãng thứ 3 (JailCoder, xCode Patcher)

Ngoài cách phải sở hữu Apple Developer ID để build được ứng dụng ta vẫn có thể build ứng dụng trên thiết bị thật mà không cần phải đăng ký tài khoản đó là sử dụng JailCoder (Xcode 4 trở xuống) hoặc xCode Patcher (Xcode 5).

Tuy nhiên, việc build ứng dụng lên thiết bị thật bằng cách dùng phần mềm của hãng thứ 3 có một số bất tiện do phải chờ phiên bản phù hợp với phiên bản Xcode được cài đặt trên máy và tính đến thời điểm hiện tại trang chủ của JailCoder đã thông báo không phát hành thêm bất kì một bản update mới nào; vì vậy đối với những ai đã update lên phiên bản Xcode 6 sẽ không thể sử dụng cách này để build ứng dụng được nữa.

Để có thể build được ứng dụng bằng JailCoder hoặc xCode Patcher thì thiết bị bắt buộc phải được JailBreak và cài đặt AppSync phù hợp với phiên bản iOS đang dùng.

a. Build ứng dụng dùng JailCoder

- Download JailCoder tại địa chỉ:
  http://khoapham.vn/nhatnghe/iphone/JailCoder.app.zip.
- Tắt Xcode và iOS Simulator đang chạy.
- Kết nối thiết bị vào máy tính.
- Khởi động JailCoder -> click Guided Patch để chuẩn bị cho quá trình patch -> click Got it để bắt đầu quá trình chuẩn bị patch.

Hình 2.32: Click Guided Patch -> Got it

– Click Certificate Root để tạo chứng chỉ root -> nhập password của máy Mac đang sử dụng (password khi đăng nhập máy) -> Modify Keychain.

Hình 2.33: Giao diện tạo Cerrtificate

– Click Certificate Private -> nhập password của máy Mac đang sử dụng (sẽ báo lỗi nhưng không ảnh hưởng đến quá trình patch) -> click OK để hoàn tất việc tạo chứng chỉ private.

Hình 2.34: Thông báo lỗi

– Click Next để hoàn tất quá trình tạo các chứng chỉ cần thiết -> click Patch My Xcode để bắt đầu quá trình patch -> nhập lại password -> OK và chờ.

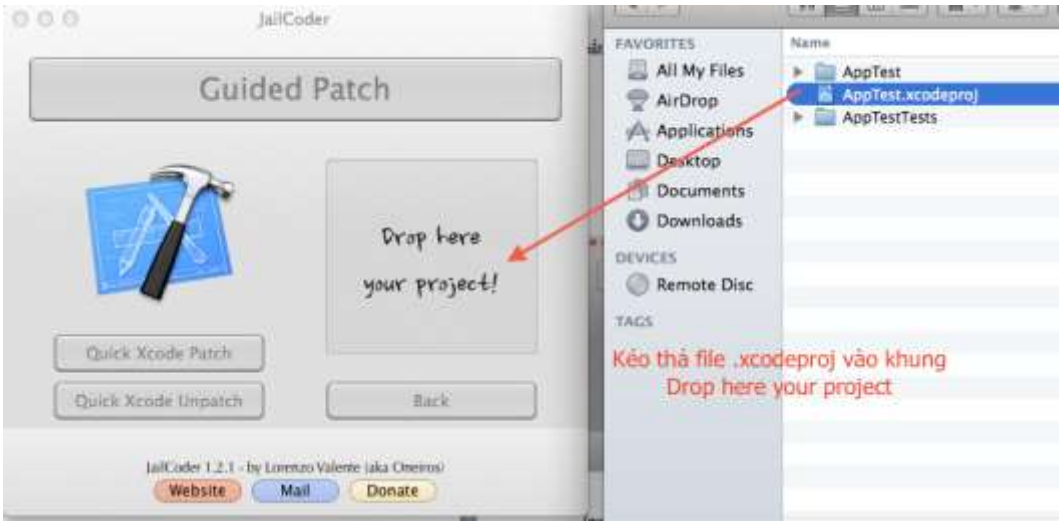
Hình 2.35: Patch my Xcode

Hình 2.36: Giao diện hiển thị khi patch thành công

– Click Back to Main Menu để trở về giao diện chính của chương trình -> tại giao diện chính, click Patch My Project để patch project ứng dụng.

Hình 2.37: Patch My Project

– Mở thư mục chứa file project ứng dụng, tìm file Xcode có đuôi .xcodeproj kéo thả vào ô Drop here your project.



Hình 2.38: Kéo thả file Xcode vào ô Drop here your project

– Mở file .xcodeproj tại vùng Deployment Info mục Deployment Target chọn version iOS nhỏ hơn hoặc bằng với version trên thiết bị.

– Trên thanh Build chọn Device là iOS Device hoặc tên thiết bị đang kết nối sau đó click Run.

– Xem ứng dụng được build trên thiết bị (lưu ý đối với những thiết bị có đặt mật khẩu khoá màn hình thì trước khi Run cần phải mở khoá màn hình nếu không sẽ không build được).

b. Build ứng dụng dùng Xcode Patcher

– Download xCode Pathcher bản tiếng việt tại địa chỉ: https://www.dropbox.com/s/6falv7kkshdyagh/xCode%205%20Unsigned%20-%20vi.app.zip?dl=0

– Tắt Xcode và iOS Simulator đang chạy.

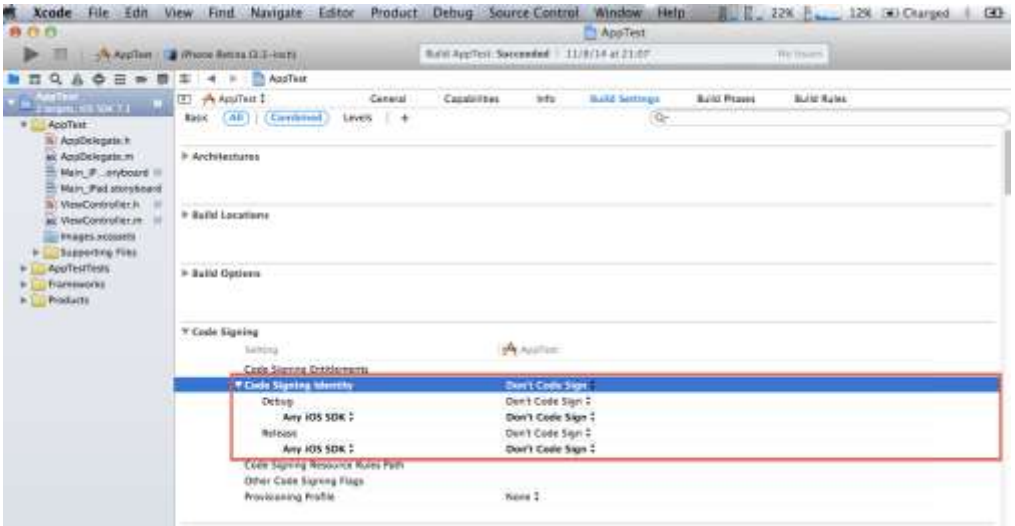
– Kết nối thiết bị vào máy tính.

– Khởi động xCode Pathcher -> click Patch xCode 5 để bắt đầu quá trình patch.

Hình 2.39: Giao diện xCode Patcher

– Mở project ứng dụng muốn build, trong phần Build Settings tìm mục Code Signing.

– Tại mục Code Signing Identity trong mục Code Signing, chỉnh tất cả lại thành Don't Code Signing.



Hình 2.40: Chỉnh sửa mục Code Signing trong project

– Trên thanh Build chọn Device là iOS Device hoặc tên thiết bị đang kết nối sau đó click Run.

2.3.2. Cài đặt và chia sẻ ứng dụng

Tương tự như việc build ứng dụng trên thiết bị, việc cài đặt và chia sẻ ứng dụng cũng có 2 cách.

2.3.2.1. Upload lên AppStore

Tương tự việc build ứng dụng trên thiết bị thật, việc upload và chia sẻ ứng dụng trên AppStore cũng cần phải có Apple Developer ID. Ngoài ra còn phải tuân thủ một số điều kiện để có thể được Apple duyệt ứng dụng và cho đăng ứng dụng trên AppStore.

Các bước để đưa ứng dụng lên AppStore cũng giống như khi build app, tuy nhiên thêm vào 2 bước là tạo file .ipa và đăng kí với Apple sau đó chờ duyệt ứng dụng. Ta có thể vào trang

https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40012582-CH1-SW1 để xem hướng dẫn chi tiết từng bước của Apple.

2.3.2.2. Upload lên các trang web khác

Cách này thường được dùng cho những người chưa có tài khoản Apple Developer ID hoặc không muốn bỏ một số tiền tương đối lớn để mua tài khoản chính thống từ Apple. Tuy nhiên các ứng dụng được upload lên các trang ngoài AppStore khi muốn cài đặt thì bắt buộc thiết bị phải được JailBeark và cài đặt AppSync phù hợp với phiên bản iOS đang sử dụng.

Nếu muốn đưa ứng dụng lên các trang web như Cydia hay AppStore.vn, ta vẫn cần phải có tài khoản của các trang web đó (tài khoản miễn phí hoặc trả phí) và cách đưa lên cơ bản là đều phải tạo được file .ipa hoặc thư mục .zip chứa app.

Ngoài ra, còn có thể tạo file .ipa sau đó chép vào thiết bị để cài đặt thông qua iTunes; cách này cũng có thể dùng để build ứng dụng tuy nhiên sẽ mất thời gian và mất công hơn so với việc sử dụng các phần mềm của hãng thứ ba.

- **Hướng dẫn tạo file .ipa**

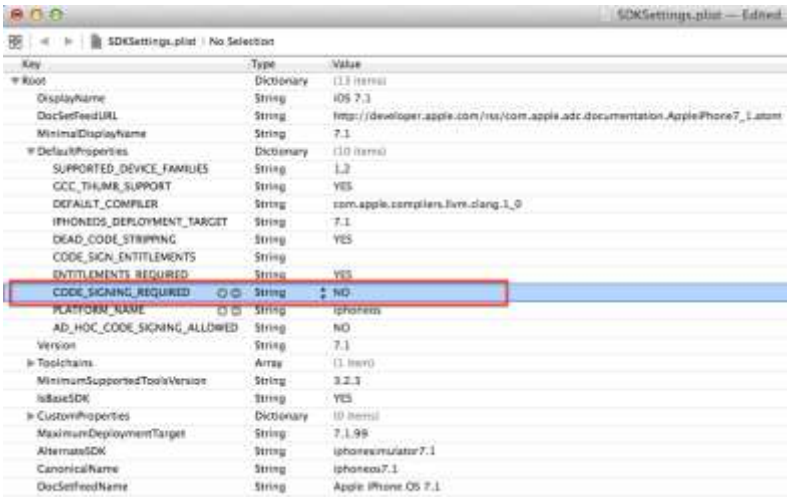
  – Mở Finder -> Applications tìm Xcode -> click chuột phải chọn Show Pakage Contents để mở đến file gốc ứng dụng.

Hình 2.41: Show thư mục gốc của Xcode

– Mở Contents -> Developer -> Platforms -> iPhoneOS.platform -> Developer -> SDKs -> iPhoneOS.sdk (hoặc iPhoneOS + phiên bản iOS + .sdk) -> tìm đến file SDKSettings.plist, mở file này trong Xcode.

Hình 2.42: Thư mục chứa file SDKSettings.plist

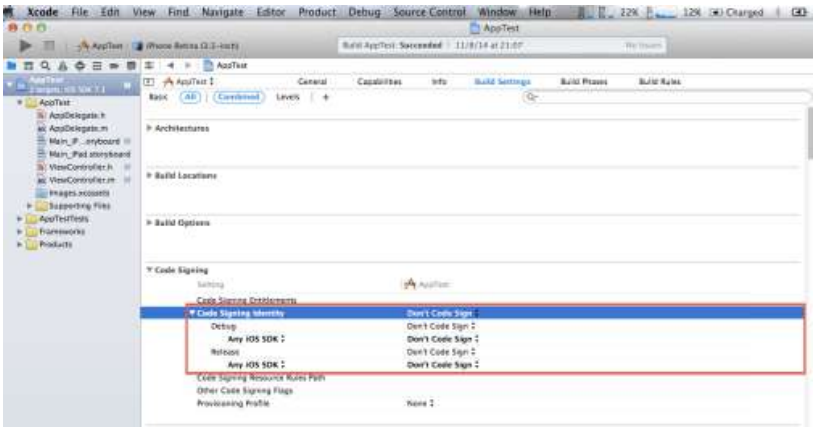
– Tại file SDKSettings.plist, tìm đến dòng DefaultProperties -> mục CODE_SIGNING_REQUIRED cột Value đổi thành NO -> lưu file lại.



Hình 2.43: Chỉnh sửa file SDKSettings

– Mở project ứng dụng muốn build trong Xcode, phần Build Settings tìm mục Code Signing.

– Tại mục Code Signing Identity trong mục Code Signing, chỉnh tất cả lại thành Don't Code Signing.



Hình 2.44: Sửa mục Code Signing của file ứng dụng

– Trên thanh Build chọn Device là iOS Device hoặc tên thiết bị đang kết nối sau đó chọn thẻ Product -> Archive.

Hình 2.45: Chọn Archive thiết bị

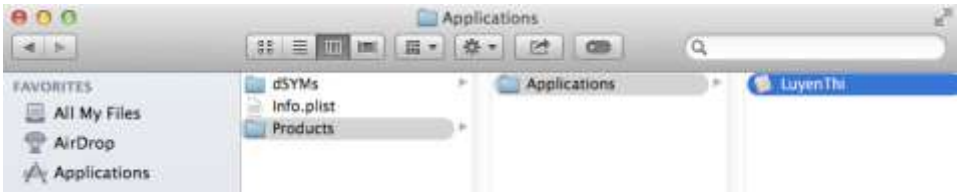
– Cửa sổ Archive hiện ra sẽ hiển thị file ứng dụng, nếu đã làm nhiều lần thì chọn file được Archive mới nhất rồi click chuột phải vào file ứng dụng chọn Show in Finder sẽ hiển thị file tên ứng dụng .xcarchive.

Hình 2.46: Cửa sổ Archive

– Click chuột phải vào file .xcarchive chọn Show Pakage Contents để mở thư mục gốc của file.

Hình 2.47: Mở thư mục gốc của file .xcarchive

– Tại thư mục Products -> Applications sẽ thấy file ứng dụng.



Hình 2.48: Thư mục lưu trữ file ứng dụng

– Mở iTunes kéo thả file ứng dụng vào iTunes để sync ra file .ipa.

– Sau khi ứng dụng đã được thêm vào iTunes, click chuột phải vào icon ứng dụng trong iTunes chọn Show in Finder để mở thư mục gốc của ứng dụng sẽ thấy file ứng dụng dạng ipa chép file đó ra nơi cần lưu trữ.

– Cuối cùng để cài đặt ứng dụng bằng file .ipa vào thiết bị, ta kết nối thiết bị vào máy tính chọn tên thiết bị trong iTunes, kéo thả file ứng dụng vào iTunes một lần nữa để thêm ứng dụng vào iTunes.

Hình 2.49: Thêm ứng dụng vào iTunes

– Chọn mục App trong thiết bị trên iTunes, tìm đến ứng dụng click Install -> Apply để bắt đầu quá trình cài đặt ứng dụng vào thiết bị.

Hình 2.50: Cài đặt ứng dụng vào thiết bị

2.4. Tiểu kết

Chương này cung cấp sơ lược về phần mềm Xcode hỗ trợ lập trình dành cho thiết bị Mac (MacBook, iMac, iPhone, iPad, …), các bước để tạo một ứng dụng dành cho thiết bị dùng hệ điều hành iOS (iPhone, iPad). Ngoài ra còn cung cấp các cách để build ứng dụng trên thiết bị thật và chia sẻ ứng dụng cho người dùng khác.

CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG LUYỆN THI LÝ THUYẾT LÁI XE

3.1. Phân tích ứng dụng

3.1.1. Nhu cầu

Hiện nay, nhu cầu ôn tập và đăng ký thi bằng lái của người dân tăng cao do việc siết chặt quá trình kiểm tra, rà soát việc chấp hành luật giao thông trong cả nước dẫn đến các trung tâm luyện thi và tổ chức thi cấp giấy phép lái xe trở nên quá tải.

Ngoài ra, người dân cũng không thể lúc nào cũng có thời gian đến các trung tâm luyện thi hoặc người dân nào cũng có máy tính, laptop để sử dụng các phần mềm ôn thi.

Chính vì những nhu cầu đó, việc phát triển một ứng dụng dùng cho các thiết bị di động thông minh như smartphone hay tablet giúp người dùng có thể tranh thủ những thời gian rảnh để ôn thi mà không mất quá nhiều thời gian và có thể sử dụng ở bất cứ đâu mà không bị ảnh hưởng quá nhiều đến những yếu tố bên ngoài.

3.1.2. Ý tưởng

Xây dựng một ứng dụng giúp người dùng luyện thi lý thuyết lái xe trên hệ điều hành iOS dùng cho iPhone, iPad.

3.1.3. Hiện trạng

3.1.3.1. Khảo sát hiện trạng

Để thi lấy giấy phép lái xe, người điều khiển phương tiện phải trải qua 2 phần thi (đối với xe máy) và 3 phần thi (đối với ô tô) là: thi lý thuyết trên máy, thi thực hành lái xe, thi đường trường (dành cho ô tô). Ngoài ra, đối với người điều khiển ô tô trước khi thi bắt buộc phải trải qua một khoá học lái xe tuỳ thuộc vào hạng bằng lái: 3 tháng (với hạng B1 và B2), 6 tháng (Hạng C), …

Phần tài liệu học dành cho 2 loại xe mô tô và ô tô là như nhau chỉ khác về số lượng câu hỏi. Bộ đề cho xe máy hạng A1, A2 là 150 câu; hạng A3, A4 là 390 câu được lấy từ bộ đề 450 câu dành cho ô tô. Các bộ đề đều bao gồm 3 phần:

– Những khái niệm, quy định giao thông của Luật giao thông đường bộ (các khái niệm, các hành vị bị nghiêm cấm, những hành vi vi phạm quy định, …), nghiệp vụ vận tải, văn hoá đạo đức, kỹ thuật lái xe và cấu tạo sửa chữa xe ô tô.

– Biển báo và ý nghĩa.

– Bài tập sa hình (những mẫu tình huống giao thông căn bản kèm hình minh hoạ).

Các hạng bằng lái:

– A1: Xe mô tô 2 bánh có dung tích xilanh từ 50 đến dưới 175 cm^3.

– A2: Xe mô tô 2 bánh có dung tích xilanh từ 175 cm^3 trở lên và các loại xe quy định cho giấy phép hạng A1.

– A3: Xe lam, mô tô 3 bánh, xích lô máy và các loại xe quy định cho giấy phép lái xe hạng A1, A2.

– A4: Máy kéo nhỏ có trọng tải đến 1000 Kg.

– B1: Ô tô chở người đến 9 chỗ ngồi, ô tô tải dưới 3500 Kg không kinh doanh vận tải.

– B2: Ô tô chở người đến 9 chỗ ngồi, ô tải đầu kéo có 1 rơ móoc dưới 3500 Kg kinh doanh vận tải và các loại xe quy định cho giấy phép hạng B1.

– C: Ô tô tải, đầu kéo có 1 rơ móoc từ 3500 Kg trở lên và các loại xe quy định cho giấy phép hạng B1, B2.

– D: Ô tô chở người từ 10 đến 30 chỗ ngồi và các loại xe quy định cho giấy phép hạng B1, B2, C.

– E: Ô tô chở người trên 30 chỗ ngồi và các loại xe quy định cho giấy phép hạng B1, B2, C, D.

– F: Ô tô tải hạng B2, C, D, E có kéo rơ móoc trên 750 Kg.

Giấy phép lái xe của tất cả các hạng trên đều được điều khiển xe máy, mô tô có dung tích xilanh dưới 50 cm^3. Những hạng bằng trên đều thuộc 1 trong 2 loại phương tiện thi là: mô tô (A1, A2, A3, A4) và ô tô (B1, B2, C, D, E, F); trong đó hạng A3, A4 bao gồm xe lam, mô tô 3 bánh, xích lô máy và xe kéo nhỏ có trọng tải dưới 1000 kg.

Đối với mỗi hạng bằng, đề thi có thể khác nhau về thời gian, số lượng câu hỏi, số câu đạt khác nhau được liệt kê trong bảng dưới:

Bảng 3.1: Khác biệt giữa các hạng bằng lái

Hạng Thi	Số Câu Hỏi	Thời Gian Làm Bài	Số Câu Đúng Tối Thiểu	Bộ Đề
A1	20	15 phút	16	150 câu
A2				
A3			18	390 câu
A4				
B1	30	20 phút	26	450 câu
B2				
C			28	
D				
E				
F				

Cơ cấu đề được quy định cho mỗi hạng bằng lái như sau:

- Mô tô, xe máy:
  + Khái niệm, quy tắc giao thông: 10 câu.
  + Biển báo và ý nghĩa biển báo: 5 câu.
  + Giải các thế sa hình: 5 câu.
- Ô tô:
  + Khái niệm, quy tắc giao thông: 9 câu.
  + Biển báo và ý nghĩa biển báo: 9 câu.
  + Giải các thế sa hình: 9 câu.
  + Nghiệp vụ vận tải: 1 câu.

+ Văn hoá, đạo đức: 1 câu.
+ Kỹ thuật lái xe và cấu tạo sửa chữa: 1 câu.

3.1.3.2. Phân tích

Mục đích chính của đề tài là xây dựng ứng dụng giúp người dùng luyện thi lý thuyết lái xe và thi thử trước khi đăng ký thi trên thực tế.

Các câu hỏi được lấy trong bộ 450 câu hỏi thi lý thuyết lái xe do bộ ban hành bao gồm các câu lý thuyết, các câu về biển báo và các câu sa hình.

Đề thi được lấy dựa vào hạng bằng lái mà người dùng chọn, các câu hỏi được lấy một cách ngẫu nhiên. Đề thi đảm bảo đúng với cơ cấu đề thi do bộ giao thông vận tải ban hành đối với từng hạng bằng lái.

3.1.4. Xác định yêu cầu chức năng

Ứng dụng gồm 4 chức năng lớn:

– Ôn thi: hỗ trợ ôn tập các câu hỏi trong bộ câu hỏi dùng để thi. Bộ câu hỏi được lấy dựa theo loại phương tiện mà người dùng chọn.

– Thi: hỗ trợ người dùng thi thử với cấu trúc, quy tắc thi tương tự như thi thực tế.

– Biển báo: hỗ trợ việc tra cứu biển báo và vạch kẻ đường giúp người dùng hiểu thêm về ý nghĩa, loại biển báo,…

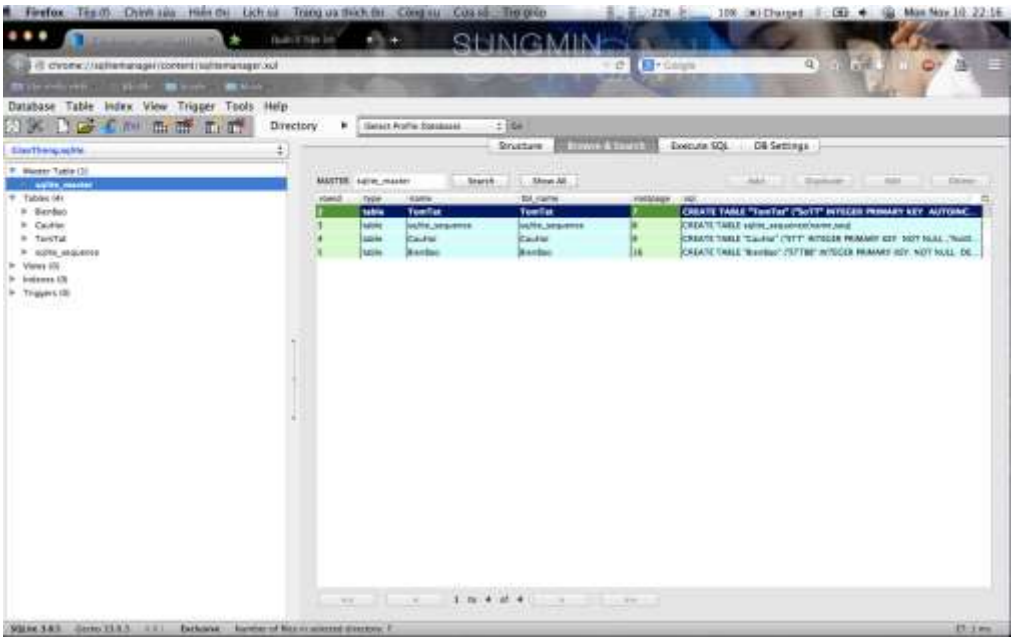
– Lý thuyết: tóm tắt một cách ngắn gọn lý thuyết lái xe, các quy định, khái niệm, quy tắc giao thông, …

3.2. Lưu trữ dữ liệu

Lưu trữ dữ liệu trong lập trình ứng dụng iOS có nhiều cách như: SQLite, CodeData, UserDefault, ….

Tuy nhiên đối với việc lưu trữ dữ liệu trong ứng dụng này, ta lựa chọn sử dụng SQLite vì nó khá gọn, nhẹ, đơn giản do chỉ dùng 1 file lưu trữ duy nhất để truy xuất dữ liệu, không cần cài đặt phức tạp và quan trọng là sử dụng ngôn ngữ SQL để truy vấn. Việc quản lý SQLite cũng khá dễ dàng thông qua plugin SQLite Manager của FireFox.

Hình 3.1: Plugin SQLite Manager trong FireFox



Hình 3.2: Giao diện SQLite Manager trong FireFox

3.2.1. Phân tích cơ sở dữ liệu

Dữ liệu được lưu trữ bao gồm các câu hỏi, biển báo, phần tóm tắt lý thuyết. Dữ liệu phải đảm bảo đáp ứng tối thiểu được những yêu cầu sau:

- Đối với câu hỏi
  - + Xác định câu hỏi dùng cho phương tiện, hạng bằng lái nào.
  - + Xác định nội dung, yêu cầu của câu hỏi.
  - + Nội dung các đáp án và đáp án đúng dành cho từng câu hỏi.
  - + Hình ảnh đối với các câu hỏi về biển báo, sa hình.

– Đối với biển báo

+ Xác định loại biển báo.

+ Xác định tên, ý nghĩa của biển báo.

+ Hình ảnh mình hoạ cho biển báo.

– Đối với phần tóm tắt lý thuyết

+ Xác định loại lý thuyết: dành cho xe máy, ô tô, …

+ Xác định nội dung phần tóm tắt lý thuyết.

3.2.2. Cấu trúc cơ sở dữ liệu

Cơ sở dữ liệu bao gồm 3 bảng với các trường dữ liệu sau:

– BienBao: STTBB, SoHieu, TenBB, YNghia, MoTa, LoaiBB, HinhBB.

– CauHoi: STT, NoiDung, CH1, CH2, CH3, CH4, LoaiCH, HinhCH, SoDACH, DACH, LoaiPT, Loai.

– TomTat: SoTT, LoaiTT, NoiDungTT, Khac.

➢ **Chú thích các trường dữ liệu:**

+ Bảng biển báo (BienBao): lưu trữ thông tin về biển báo.

- STTBB: số thứ tự của biển báo, số này là khoá chính cho bảng và là giá trị tự tăng.
- SoHieu: số hiệu của biển báo theo quy chuẩn của Bộ Giao Thông Vận Tải.
- TenBB: tên của biển báo.
- YNghia: ý nghĩa của biển báo.
- MoTa: mô tả về màu sắc, hình dáng của biển báo.
- LoaiBB: loại biển báo (cấm, hiệu lệnh, chỉ dẫn, nguy hiểm, phụ, vạch kẻ đường).
- HinhBB: hình ảnh biển báo.

+ Bảng câu hỏi (CauHoi): lưu trữ thông tin các câu hỏi trong bộ 450 câu hỏi.

- STT: số thứ tự của câu hỏi, số này là khoá chính và là giá trị tự tăng.

- NoiDung: nội dung câu hỏi.
- CH1, CH2, CH3, CH4: nội dung các đáp án.
- LoaiCH: loại câu hỏi (lý thuyết, sa hình, biển báo).
- HinhCH: hình ảnh của những câu hỏi về biển báo, sa hình.
- SoDACH: số đáp án đúng cho câu hỏi.
- DACH: đáp án đúng của câu hỏi.
- LoaiPT: câu hỏi dành cho loại phương tiện nào.
- Loai: câu hỏi thuộc loại nào trong phần câu hỏi lý thuyết (khái niệm quy tắc giao thông, nghiệp vụ vận tải, văn hoá đạo đức, kỹ thuật lái xe, cấu tạo sửa chữa).

+ Bảng tóm tắt (TomTat): lưu trữ lý thuyết được tóm tắt.

- SoTT: số thứ tự, là khoá chính và là giá trị tự tăng.
- LoaiTT: loại lý thuyết được tóm tắt (biển báo, sa hình, quy định chung, quy định đối với phương tiện, nghiệp vụ vận tải, kỹ thuật lái xe, quy tắc giao thông, điều kiện ký thuật xe, lý thuyết luật).
- NoiDungTT: nội dung phần lý thuyết được tóm tắt.
- Khac: các đề mục nhỏ hơn trong các loại lý thuyết được tóm tắt.

3.3. Định hướng người dùng

Sau khi sử dụng ứng dụng, người dùng đạt được một số mục tiêu sau:

– Thi thử với cấu trúc đề thi tương tự như thi thực tế.

– Ôn thi các câu hỏi trong bộ đề 450 câu.

– Nhận biết được một số biển báo hiệu đường bộ.

– Biết thêm một số kiến thức cơ bản về luật giao thông đường bộ.

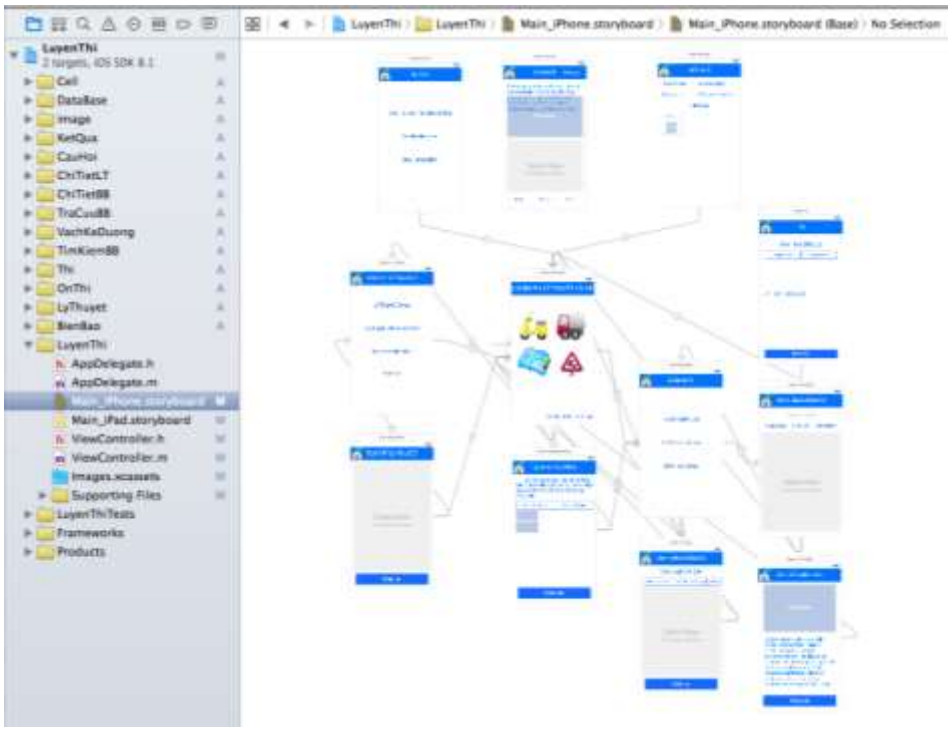
3.4. Xây dựng ứng dụng

3.4.1. Thiết kế giao diện và xây dựng cơ sở dữ liệu cho ứng dụng

3.4.1.1. Thiết kế giao diện

Ứng dụng cần thiết kế 12 viewController khác nhau tương ứng với 12 giao diện bao gồm: giao diện chính, ôn thi, thi, câu hỏi, kết quả, tóm tắt lý thuyết, chi tiết

lý thuyết, biển báo, tìm kiếm biển báo, tra cứu biển báo, vạch kẻ đường và chi tiết biển báo.



Hình 3.3: Các ViewController tương ứng với các giao diện của ứng dụng

Mỗi viewController được thiết kế tuỳ theo chức năng của giao diện, ngoài ra cần tạo 2 file .m và .h tương ứng để quản lý các viewController.

Ví dụ thiết kế giao diện vạch kẻ đường; ta làm tương tự với các giao diện khác.

a. Giao diện

Với giao diện vạch kẻ đường, ta cần dùng các control sau:

– 2 label: một dùng để hiển thị tên của giao diện ở đây là "Vạch Kẻ Đường" và một dùng để hiển thị khái niệm về vạch kẻ đường.

– 1 segmentedControl: để người dùng lựa chọn loại vạch kẻ đường là: vạch kẻ ngang hay vạch kẻ đứng.

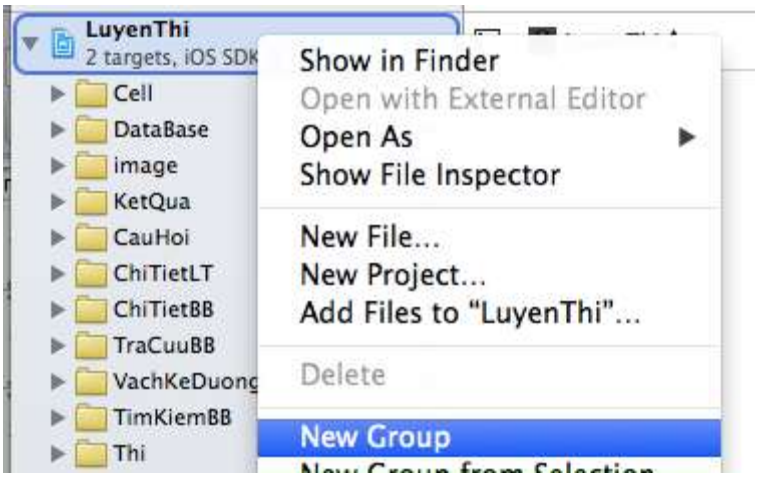
– 1 collectionView: để hiển thị hình ảnh của các vạch kẻ đường. Trong collectionView có thêm một control con là collectionViewCell, trong control này ta thêm vào 1 imageView để hiển thị hình.

– 2 button: một để quay lại giao diện trước đó là giao diện Biển Báo và một để quay lại giao diện chính.

Hình 3.4: Thiết kế giao diện Vạch Kẻ Đường

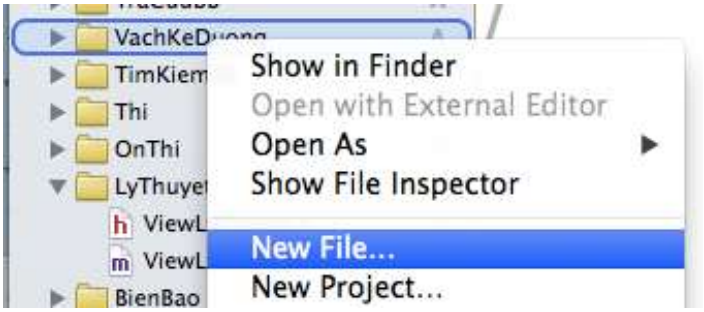
b. Tạo file quản lý

– Trong Xcode, nhấp chuột phải vào project -> New Group rồi đặt tên, ở đây là VachKeDuong.



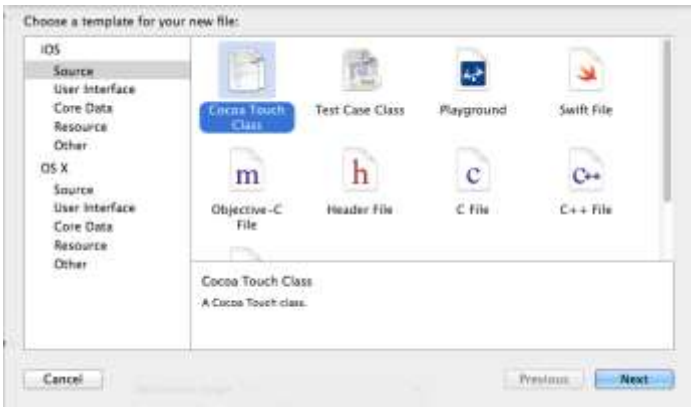
Hình 3.5: Thêm thư mục mới trong project

– Nhấp chuột phải vào thư mục vừa tạo -> New File… để tạo file.



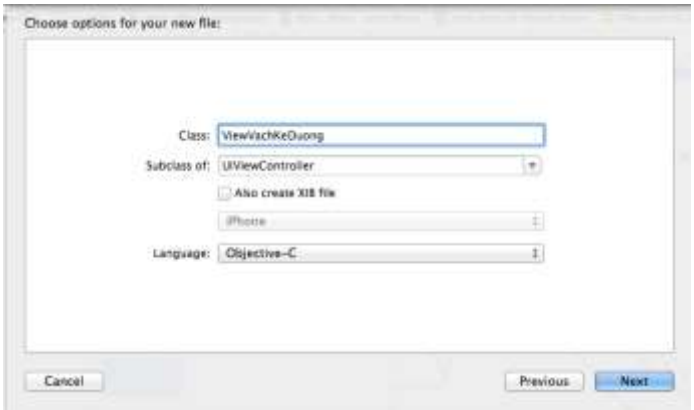
Hình 3.6: Thêm file vào thư mục vừa tạo

– Cửa sổ xuất hiện, chọn CoCoa Touch Class (template này dùng để tạo 2 file .m và .h)-> Next để chọn template cho file.



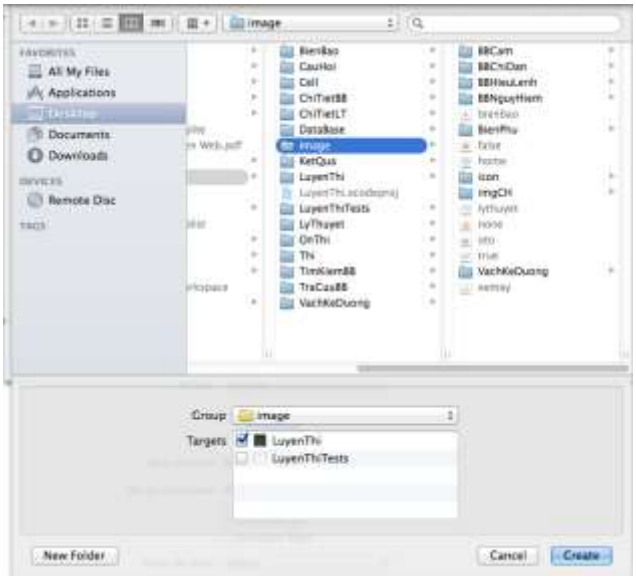
Hình 3.7: Chọn teamplate CoCoa Touch Class

– Tại ô Class đặt tên cho file sắp tạo (ViewVachKeDuong), ô Subclass of để chọn loại Controller thích hợp (UIViewController), ô Language để chọn ngôn ngữ để viết ứng dụng là Objective C hay Swift (ô này chỉ có trong Xcode 6) -> Next để xác nhận các thông tin đã nhập.



Hình 3.8: Cấu hình thông tin cho file

– Hiển thị giao diện chọn nơi lưu file -> Create để tạo file.



Hình 3.9: Chọn nơi lưu file

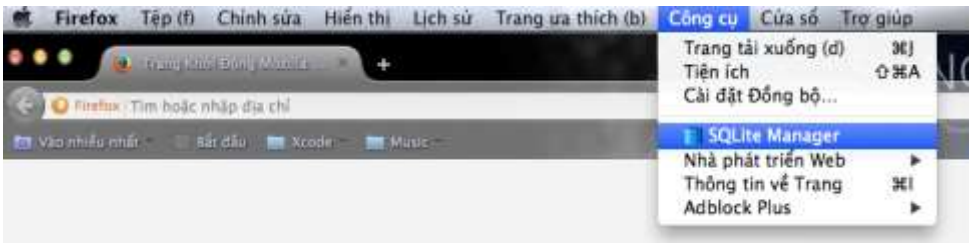
– Mở file thiết kế (.storyboard), chọn giao diện vạch kẻ đường vừa thiết kế -> gõ tên file vừa tạo vào ô Class trong phần Custom Class để file vừa tạo có thể quản lý được view.

Hình 3.10: Cấu hình để file vừa tạo có thể quản lý view

3.4.1.2. Xây dựng cơ sở dữ liệu

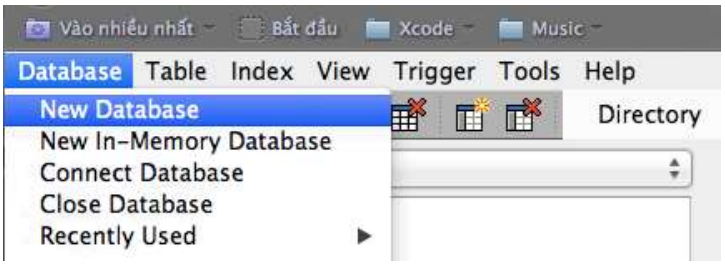
Các bước để xây dựng cơ sở dữ liệu:

– Mở trình duyệt FireFox, chọn thẻ Công Cụ (Tools) -> chọn SQLite Manager.

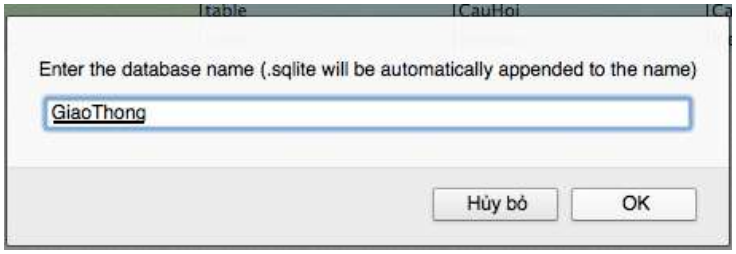


Hình 3.11: Mở công cụ SQLite Manager

– Trong giao diện SQLite Manager, chọn Database -> New Database để tạo cơ sở dữ liệu mới.

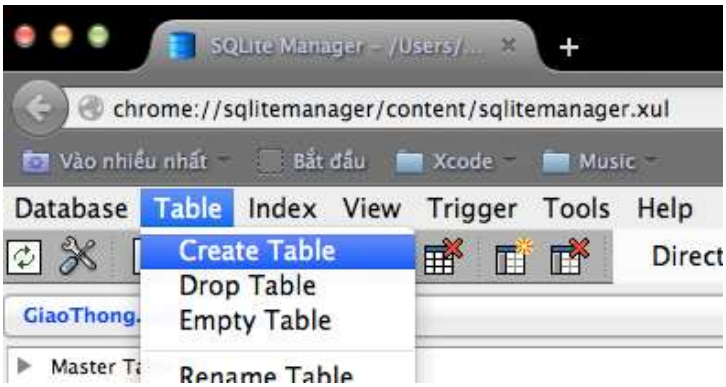


Hình 3.12: Tạo database mới



Hình 3.13: Đặt tên cho database

– Sau khi tạo xong cơ sở dữ liệu, chọn Table -> New Table để tạo các bảng với các trường dữ liệu như đã phân tích tại mục 3.2.2 cấu trúc cơ sở dữ liệu.

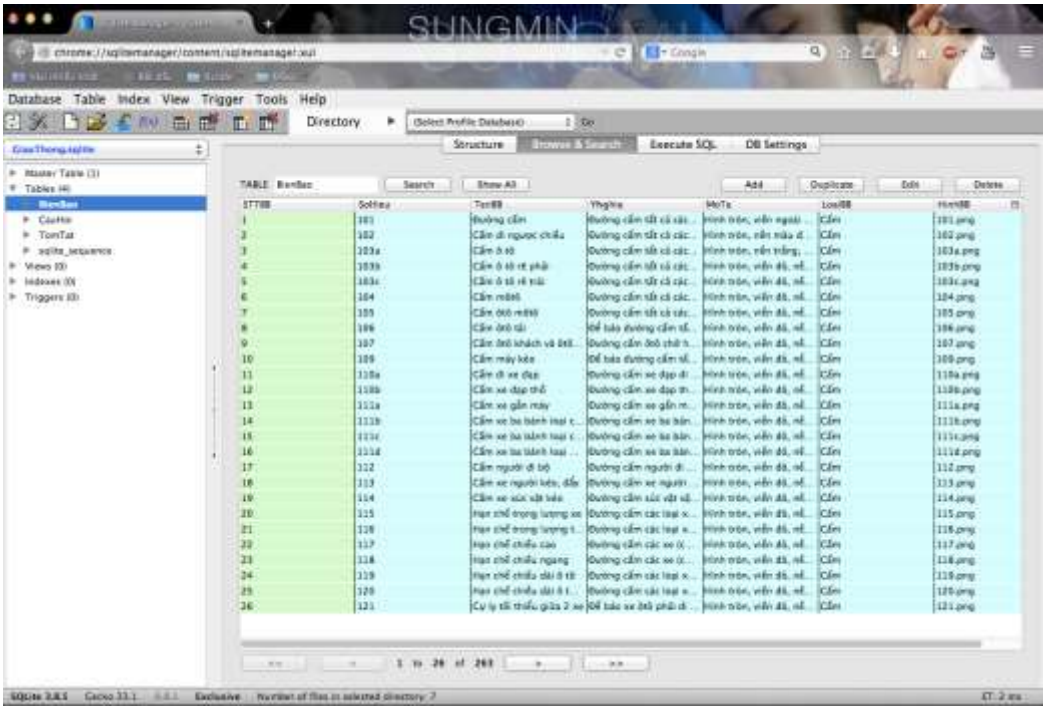


Hình 3.14: Tạo bảng mới

Hình 3.15: Thêm các trường dữ liệu cho bảng

– Cuối cùng nhập dữ liệu vào các bảng để hoàn tất việc xây dựng cơ sở dữ liệu.

Hình 3.16: Giao diện nhập dữ liệu vào bảng



Hình 3.17: Bảng đã được nhập dữ liệu

3.4.2. Liên kết giao diện và khai báo các đối tượng

3.4.2.1. Liên kết giao diện

Thực hiện liên kết giao diện để có thể chuyển đổi từ giao diện này sang giao diện khác thông qua các sự kiện của đối tượng được chọn dùng để liên kết (button, switch, segmentedcontrol, …). Liên kết giao diện gồm 2 cách: liên kết trực tiếp trên giao diện và liên kết bằng code.

Đối với ứng dụng này, ta sử dụng cả 2 loại liên kết giao diện:

a. Liên kết trực tiếp trên giao diện

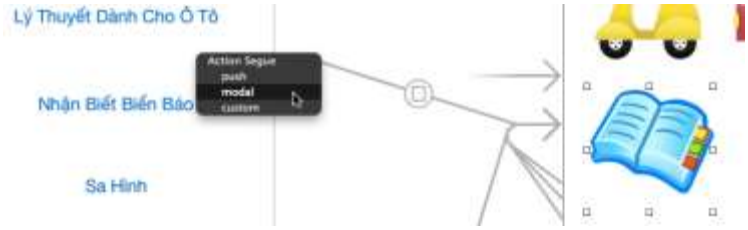
Liên kết trực tiếp trên giao diện dùng cho button hình ngôi nhà để quay về giao diện chính (trừ button tại giao diện câu hỏi), button quay lại giao diện trước giao diện hiện tại (trừ button tại giao diện chi tiết biển báo), button lý thuyết (hình cuốn sách), button biển báo (hình biển báo) và các button tại giao diện biển báo.

Để liên kết trực tiếp trên giao diện ta thực hiện các bước sau:

– Nhấn kết hợp phím control và nhấp chuột vào đối tượng muốn tạo liên kết (ở đây là button lý thuyết), kéo thả sang giao diện muốn liên kết tới (ViewLyThuyet).

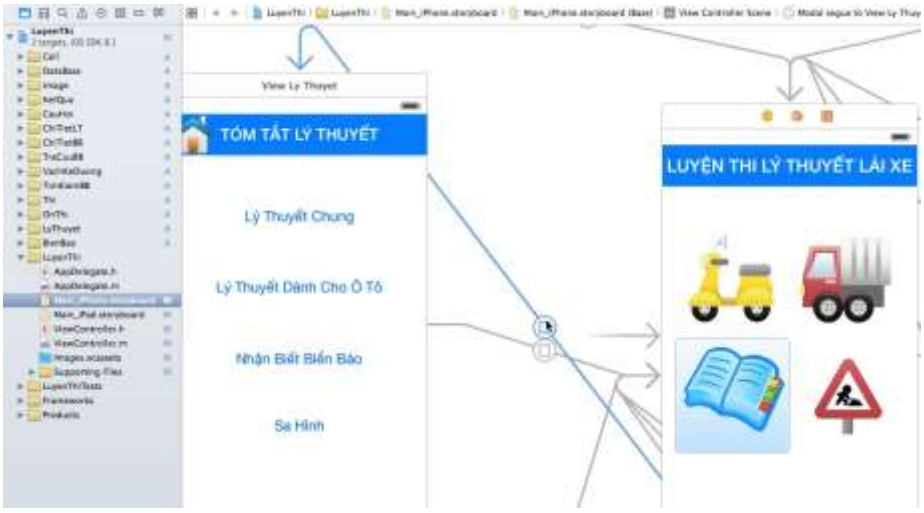
Hình 3.18: Tạo liên kết giao diện

– Xcode sẽ hiển thị danh sách các kiểu liên kết giao diện để lựa chọn như push, modal hoặc custom; riêng đối với kiểu liên kết là push chỉ sử dụng cho các view có dạng view cha con (page viewController, table viewController, …).



Hình 3.19: Danh sách các kiểu liên kết giao diện

– Sau khi liên kết xong sẽ xuất hiện một sợi dây liên kết nối từ đối tượng được chọn tới giao diện được liên kết; làm tương tự cho các button khác.



Hình 3.20: Liên kết giao diện đã được tạo

b. Liên kết bằng code

Liên kết giao diện bằng code được dùng khi cần chuyển dữ liệu từ giao diện này (giao diện nguồn) sang giao diện khác (giao diện đích) hoặc trong sự kiện của đối tượng tạo liên kết còn có các sự việc cần phải thực hiện trước khi chuyển giao diện.

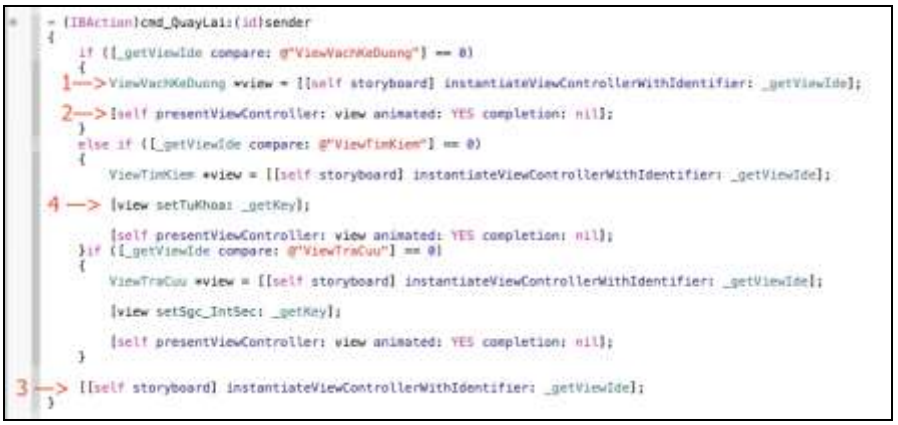
Ví dụ dưới đây hướng dẫn liên kết code từ button "Quay Lại" của giao diện chi tiết biển báo (ViewChiTietBB) đến giao diện vạch kẻ đường.

– Khai báo button "Quay Lại" dạng action để có thể tương tác với button.

Hình 3.21: Khái báo sự kiện cho button "Quay Lại"

– Import thư viện của giao diện vạch kẻ đường vào file .h của giao diện chi tiết biển báo (ViewChiTietBB.h).

– Tại file .m của giao diện chi tiết biển báo (ViewChiTietBB.m), ta tìm đến sự kiện của button "Quay Lại" để viết code liên kết. Do button "Quay Lại" tại giao diện chi tiết biển báo dùng để quay lại giao diện vạch kẻ đường, tìm kiếm và tra cứu biển báo nên cần có biến _getViewIde (biến này được truyền dữ liệu từ một trong 3 giao diện trên) để xác định được biển báo này thuộc loại giao diện nào đưa tới.



Trong đó:

+ Lệnh số 1 để khai báo biến với kiểu dữ liệu là kiểu giao diện đích (ViewVachKeDuong, ViewTraCuu, ViewBienBao).

+ Lệnh số 2 dùng để truyền dữ liệu.

+ Lệnh thứ 3 dùng để chuyển giao diện.

+ Đối với lệnh thứ 4 (lệnh của 2 view tra cứu và tìm kiếm) là lệnh gán dữ liệu từ giao diện chi tiết biển báo đến 1 trong hai giao diện trên.

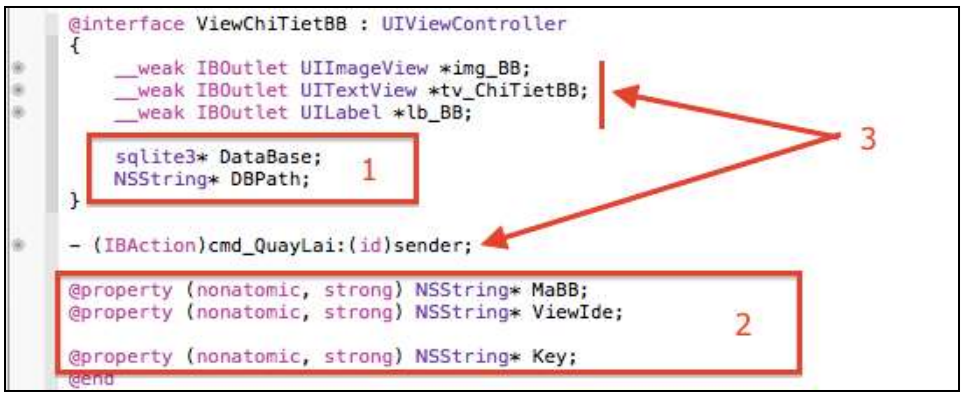
➢ Với biến truyền vào dòng lệnh số 1 và số 3 có giá trị được gán là Storyboard ID của giao diện đích (ViewVachKeDuong, ViewTraCuu, ViewBienBao).

Storyboard ID là một chuỗi ký tự dùng để xác định giao diện người dùng tự tạo tại mục Identity của giao diện.

Hình 3.22: Storyboard ID của giao diện

3.4.2.2. Khai báo các đối tượng

Khai báo các đối tượng sử dụng trong file .h để có thể sử dụng các đối tượng trong quá trình viết code, ngoài ra còn khai báo các biến cục bộ dùng cho toàn bộ giao diện và biến dùng để truyền dữ liệu từ giao diện này sang giao diện khác. Dưới đây là ví dụ về file . h của giao diện chi tiết biển báo (ViewChiTietBB.h).



Hình 3.23: Code khai báo trong file .h

Trong đó:

– Khung số 1 là khai báo các biến cục bộ dùng trong giao diện chi tiết biển báo, các biến này được đặt trong dấu ngoặc nhọn ({}).

– Khung số 2 là khai báo các biến dùng để truyền dữ liệu (biến get_set). Đối với loại biến này để có thể sử dụng được ta cần phải định nghĩa lại nó trong file .m; dưới dòng @implementation ta thêm các dòng @synthesize tương ứng với số biến được khai báo trong file .h.

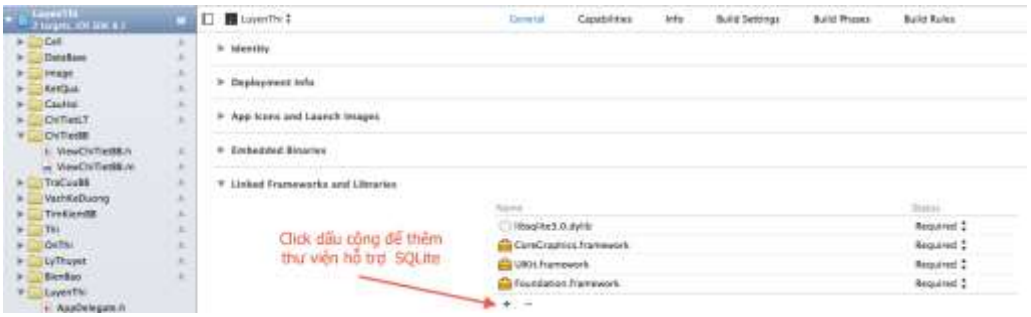
```
@implementation ViewChiTietBB
@synthesize MaBB = _getMaBB;
@synthesize ViewIde = _getViewIde;

@synthesize Key = _getKey;
```

– Các câu lệnh được đánh số 3 là lệnh khai báo các đối tượng (control) để có thể tương tác với chúng.

3.4.3. Liên kết cơ sở dữ liệu

Để ứng dụng có thể thao tác với cơ sở dữ liệu được lưu trữ bằng SQLite cần phải thêm thư viện hỗ trợ vào project như sau: trong phần General, mục Linked Frameworks and Libraries click vào dấu cộng sau đó gõ libsqlite3.0.dylib; cuối cùng click Add để thêm thư viện hỗ trợ truy cập cơ sở dữ liệu SQLite.



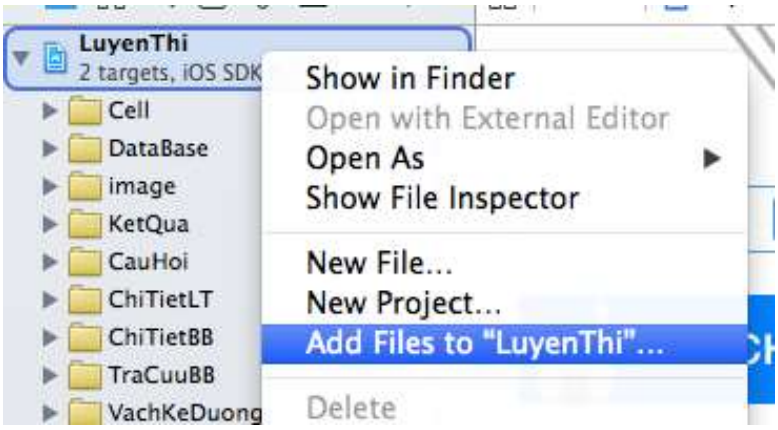
Hình 3.24: Thêm thư viện hỗ trợ SQLite

Các bước liên kết và truy xuất cơ sở dữ liệu:

– Import thư viện sqlite3.h vào file .h cần truy cập dữ liệu để có thể sử dụng được các hàm hỗ trợ việc truy xuất dữ liệu trong SQLite.

```
#import <sqlite3.h>
```

– Thêm cơ sở dữ liệu (file .sqlite) vào project.



Hình 3.25: Thêm database vào project

– Tạo đối tượng SQLite bằng cách khái báo biến có kiểu dữ liệu sqlite3 để lưu trữ database và biến có kiểu NSString để lưu trữ đường dẫn trong file .h cần truy xuất dữ liệu.

```
sqlite3* DataBase;
NSString* DBPath;
```

– Viết code để lấy đường dẫn đến file cơ sở dữ liệu cần dùng trong hàm viewDidLoad và lưu trữ đường dẫn vào biến NSString vừa tạo ở trên.

```
NSString *docdir;
docdir = NSSearchPathForDirectoriesInDomains(NSDocumentationDirectory, NSUserDomainMask, YES)[0];
DBPath = [[NSString alloc] initWithString: [docdir stringByAppendingString: @"DB_GT.sqlite"]];
NSFileManager *filemanager = [NSFileManager defaultManager];

if ([filemanager fileExistsAtPath: DBPath] == NO) {
    NSString *databasePath = [[[NSBundle mainBundle] resourcePath] stringByAppendingPathComponent: @"DB_GT.sqlite"];
    [filemanager copyItemAtPath: databasePath toPath: DBPath error: nil];
}
```

Hình 3.26: Code lấy đường dẫn đến database

– Tại sự kiện của đối tượng muốn truy xuất dữ liệu:

```
const char *path = [DBPath UTF8String]; <—1

if (sqlite3_open(path, &DataBase) == SQLITE_OK) <—2
{
    sqlite3_stmt *statement; <—3

    NSString *query = [NSString stringWithFormat: @"SELECT * FROM TomTat WHERE LoaiTT = \"%@\"", _getLoaiLT]; <— 4
    const char* sql_query = [query UTF8String]; <— 5

    if (sqlite3_prepare_v2(DataBase, sql_query, -1, &statement, nil) == SQLITE_OK) <— 6
    {
        while (sqlite3_step(statement) == SQLITE_ROW) <— 7
        {
            [MangTT addObject: [NSString stringWithUTF8String:(char*) sqlite3_column_text(statement, 2)]]; <— 8
        }
    }
    else
        NSLog(@"khong co du lieu");
}
else
    NSLog(@"Loi");
```

- Trong đó:

+ Lệnh số 1: khai báo biến có kiểu dữ liệu const char để lưu trữ đường dẫn database đã được chuẩn hoá sang định dạng UTF8.

+ Lệnh số 2: mở kết nối đến cơ sở dữ liệu bằng hàm sqlite3_open() và dùng biến SQLITE_OK để kiểm tra việc kết nối có thành công hay không.

+ Lệnh số 3: khai báo biến kiểu sqlite_stmt để khởi tạo và lưu trữ câu lệnh truy vấn.

+ Lệnh số 4: khái báo biến string chứa câu lệnh SQL. Câu lệnh SQL trên hình dùng để liệt kê tất cả các cột dữ liệu trong bảng TomTat với điều kiện là LoaiTT phải bằng với chuỗi giá trị được truyền vào.

+ Lệnh số 5: chuẩn hoá câu lệnh SQL sang định dang UTF8.

+ Lệnh số 6: truyền biến kiểu sqlite_stmt vào hàm sqlite3_prepare_v2() để thực thi; hàm này cũng dùng biến SQLITE_OK để kiểm tra câu lệnh truy vấn có thực hiện hay không.

+ Lệnh số 7: nếu câu lệnh truy vấn được thực hiện, dùng hàm sqlite3_step() kết hợp với biến SQLITE_ROW để kiểm tra và chuyển từng dòng dữ liệu.

+ Lệnh số 8: gán giá trị của dòng dữ liệu vào mảng ứng với cột được chọn (cột số 2).

3.5. Tiểu kết

Chương này trình bày hiện trạng, những phân tích về ứng dụng, cơ sở dữ liệu và cách thức lưu trữ dữ liệu. Ngoài ra còn nêu các chức năng của ứng dụng và định hướng người dùng cho ứng dụng và tóm tắt các bước thiết kế ứng dụng.

CHƯƠNG 4: GIAO DIỆN ỨNG DỤNG

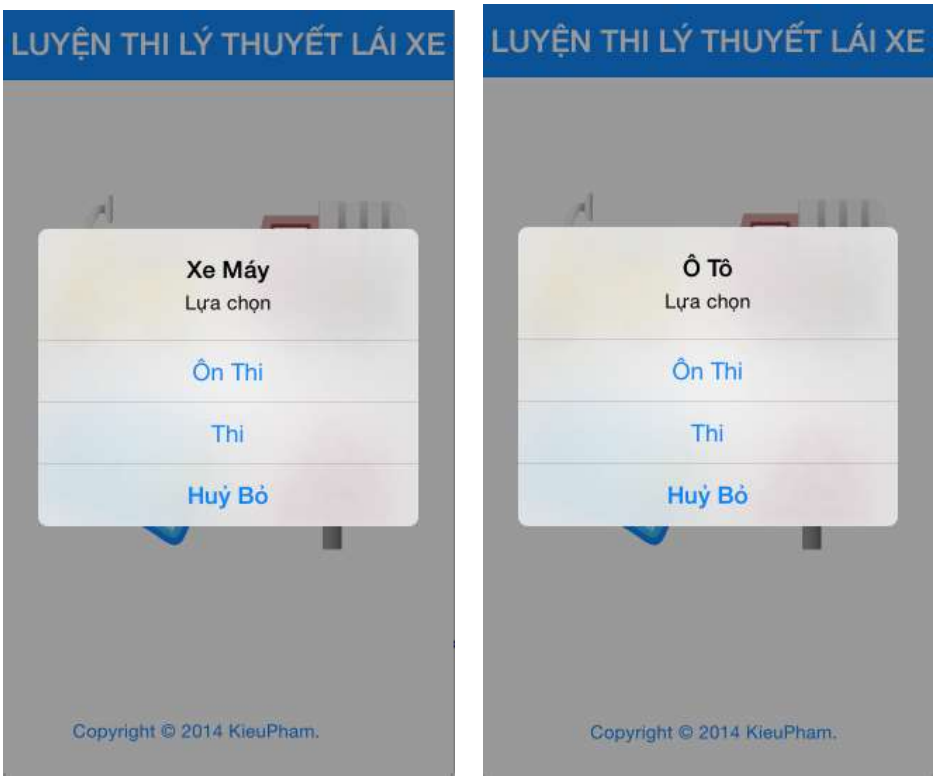
4.1. Giao diện chính

– Khi người dùng chọn vào icon ứng dụng sẽ hiển thị giao diện chính của ứng dụng.

– Tại giao diện chính, người dùng lựa chọn các button sẵn có của ứng dụng sao cho phù hợp với yêu cầu của bản thân như: xe máy, ô tô, tóm tắt lý thuyết hay biển báo.

Hình 4.1: Giao diện chính của ứng dụng

– Đối với 2 button ô tô và xe máy: khi người dùng chọn vào 1 trong 2 loại phương tiện trên sẽ hiển thị một thông báo cho người dùng chọn chức năng muốn dùng là ôn thi hay thi.



Hình 4.2: Giao diện lựa chọn chức năng trong 2 button Xe máy, Ô tô

4.2. Giao diện ôn thi

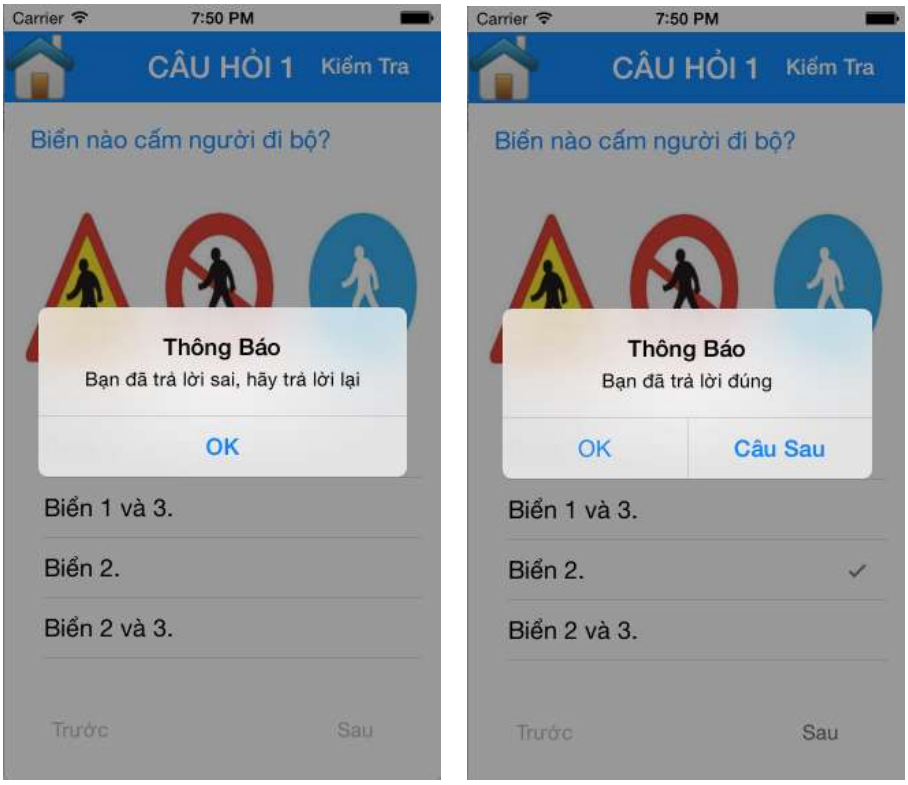
– Khi người dùng chọn chức năng "Ôn Thi", giao diện ôn thi sẽ hiển thị các loại câu hỏi (câu hỏi về quy tắc giao thông, câu hỏi về biển báo và câu hỏi về sa hình) để người dùng lựa chọn.

Hình 4.3: Giao diện ôn thi

– Với mỗi loại câu hỏi sẽ hiển thị các câu hỏi tương ứng.

– Với mỗi câu hỏi sẽ có các đáp án, người dùng lựa chọn đáp án sau đó chọn Kiểm Tra để kiểm tra xem đáp án mình chọn có đúng hay không. Nếu chọn sai, người dùng bắt buộc phải chọn lại cho đến khi kiểm tra là đúng thì mới có thể qua câu tiếp theo.

Hình 4.4: Giao diện câu hỏi trong chức năng ôn thi

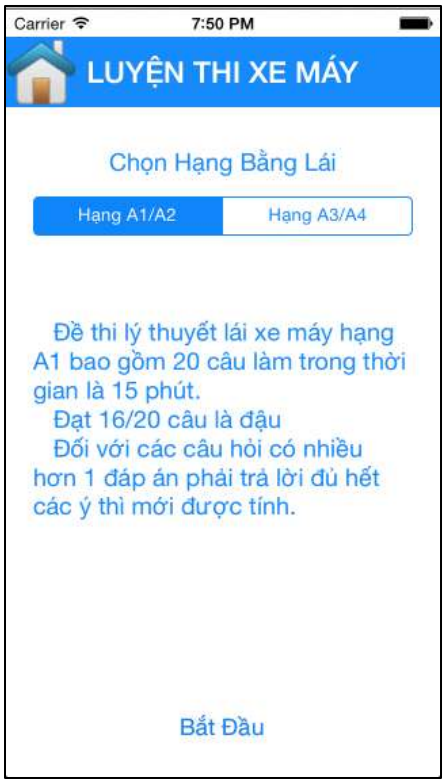


Hình 4.5: Giao diện thông báo khi kiểm tra đáp án

4.3. Giao diện thi

– Khi người dùng chọn chức năng "Thi", giao diện thi sẽ hiển thị các hạng bằng tương ứng với phương tiện thi. Người dùng lựa chọn hạng bằng nào sẽ hiển thị thông tin về hạng bằng đó như số câu hỏi, số câu đạt tối thiểu, thời gian làm bài giống như khi thi thực tế.

– Khi người dùng chọn "Bắt Đầu" sẽ xuất hiện giao diện câu hỏi và bắt đầu tính giờ.



Hình 4.6: Giao diện thi

– Với mỗi câu hỏi sẽ có các đáp án trả lời, nguòi dùng lựa chọn đáp án đúng. Đối với những câu hỏi có nhiều hơn 1 đáp án đúng, người dùng phải chọn đủ và chính xác thì mới được tính là 1 câu đúng.

– Người dùng có thế bỏ qua các câu hỏi khi chưa nghĩ ra đáp án để làm câu tiếp theo; sau đó có thể quay trở lại để hoàn thành các câu đã bỏ qua hoặc để kiểm tra lại bài làm.

Hình 4.7: Giao diện câu hỏi trong chức năng thi

– Người dùng có thể nộp bài trước khi hết giờ nếu đã hoàn thành xong bài thi. Sau khi nộp bài sẽ hiển thị giao diện "Kết Quả" thông báo tổng số câu hỏi, số câu đúng, số câu sai, số câu chưa làm, kết quả thi (Đạt hay Không Đạt) và bảng liệt kê các câu hỏi.

– Nếu hết thời gian làm bài mà người dùng vẫn chưa hoàn thành xong bài thi hoặc đã hoàn thành mà chưa nộp bài thì sẽ xuất hiện thông báo hết giờ và bắt buộc người dùng phải nộp bài; sau đó sẽ hiển thị giao diện "Kết Quả" để người dùng kiểm tra.

– Đối với bảng liệt kê câu hỏi ở giao diện "Kết Quả", người dùng có thể lựa chọn bất kì một câu hỏi nào trong bài thi để xem đáp án mình đã chọn và đáp án đúng.

Hình 4.8: Giao diện kết quả

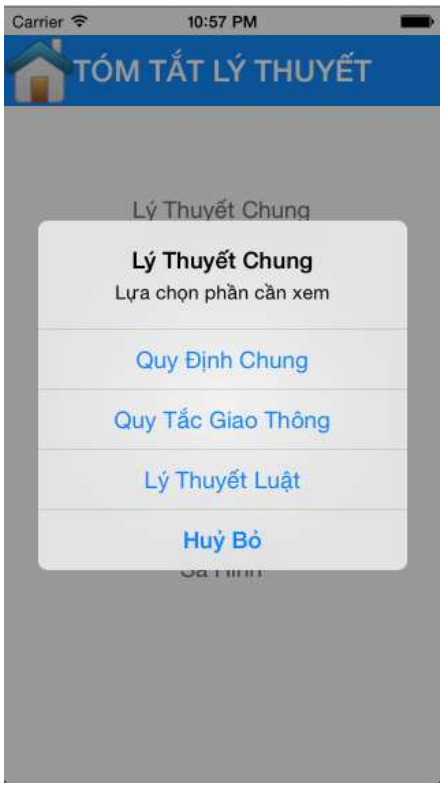
4.4. Giao diện tóm tắt lý thuyết

– Khi người dùng chọn button này sẽ hiển thị giao diện tóm tắt lý thuyết gồm các phần khác nhau: Lý Thuyết Chung, Lý Thuyết Dành Cho Ô Tô, Sa Hình và Nhận Biết Biển Báo.

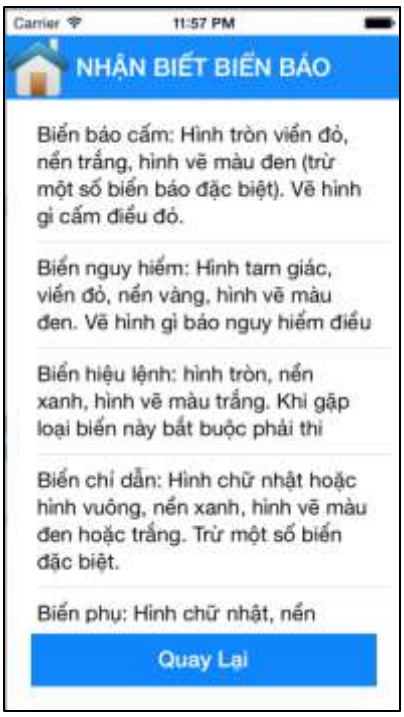
– Mỗi phần được chọn sẽ hiển thị giao diện chi tiết tóm tắt lý thuyết liên quan đến phần đó.

– Giao diện chi tiết tóm tắt lý thuyết sẽ hiển thị các mục tóm tắt theo bảng.

Hình 4.9: Giao diện tóm tắt lý thuyết



Hình 4.10: Giao diện lựa chọn phần lý thuyết



Hình 4.11: Giao diện chi tiết tóm tắt lý thuyết theo từng phần

4.5. Giao diện biển báo

– Khi chọn button này sẽ hiển thị giao diện cho người dùng lựa chọn như: Tra Cứu Biển Báo, Tìm Kiếm Biển Báo và Vạch Kẻ Đường.

Hình 4.12: Giao diện biển báo

– Phần tra cứu biển báo: hiển thị giao diện tra cứu theo loại biển báo (cấm, nguy hiểm, hiệu lệnh, chỉ dẫn, phụ), người dùng chọn loại biển báo cần tra cứu sẽ hiển thị các biển báo thuộc loại biển báo đó vào bảng bên dưới gồm tên và hình biển báo.

Hình 4.13: Giao diện tra cứu biển báo

– Phần tìm kiếm biển báo: hiển thị giao diện tìm kiếm cho phép người dùng tìm kiếm các biển báo theo từ khoá. Người dùng nhập từ khoá vào ô trống sau đó chọn “Tìm Kiếm” hệ thống sẽ tìm kiếm dựa trên tên, mô tả và ý nghĩa của biển báo ứng với từ khoá; sau đó hiển thị các biển báo vào bảng bên dưới gồm tên và hình cửa biển báo.

Hình 4.14: Giao diện tìm kiếm biển báo

– Phần vạch kẻ đường: hiển thị giao diện hiển thị vạch kẻ đường theo 2 loại (vạch kẻ đứng và vạch kẻ ngang). Người dùng lựa chọn lại vạch kẻ đường sẽ hiển thị hình ảnh của các vạch kẻ đường loại tương ứng vào bảng bên dưới.

Hình 4.15: Giao diện vạch kẻ đường

– Đối với các biển báo được hiển thị trong các giao diện trên, khi người dùng chọn vào biển báo nào sẽ hiển thị giao diện chi tiết về biển báo đó như: số hiệu, tên biển báo, ý nghĩa, mô tả và hình biển báo.

Hình 4.16: Giao diện chi tiết biển báo

PHẦN KẾT LUẬN

Đề tài nghiên cứu với mục tiêu xây dựng được một ứng dụng giúp người dùng luyện thi lý thuyết lái xe trên hệ điều hành iOS (dùng cho iPhone, iPad); với mục tiêu trên em đã nghiên cứu và đạt được kết quả sau:

Thứ nhất là tìm hiểu về ngôn ngữ Objective C, phần mềm Xcode, cách tạo một ứng dụng, cấu trúc của một ứng dụng, cách test ứng dụng trên thiết bị thật và chia sẻ ứng dụng.

Thứ hai là xây dựng được ứng dụng luyện thi lý thuyết lái xe trên hệ điều hành iOS (dành cho iPhone, iPad) giúp người dùng luyện thi và thi thử trước khi đăng ký thi trên thực tế, giảm bớt thời gian khi đến các trung tâm luyện thi và tiện lợi khi có thể dùng ở bất kì đâu với chỉ một chiếc iPhone hoặc iPad. Ngoài ra, ứng dụng còn giúp người dùng nâng cao kiến thức, khuyến khích người dùng tìm hiểu về luật giao thông nhằm giảm thiểu tai nạn giao thông bằng cách rút gọn, tóm tắt một cách ngắn ngọn những phần lý thuyết quan trọng và được phân chia làm nhiều loại khác nhau.

Tuy nhiên, với thời gian có hạn và lượng kiến thức về các vấn đề còn ít nên ứng dụng vẫn chưa được hoàn chỉnh, còn hạn chế nhiều về giao diện và các chức năng:

- Về giao diện: giao diện còn quá đơn giản, chưa tối ưu được việc thiết kế dẫn đến chưa tận dụng hết được màn hình hiển thị.

- Về chức năng: còn nhiều hạn chế dẫn đến việc sai lệch trong quá trình sử dụng như: đáp án bị đánh trước hay việc tra cứu biển báo còn bị lag, ...

- Về việc phát triển, nâng cấp ứng dụng: do thiết kế cơ sở dữ liệu chưa được tốt dẫn đến việc update cơ sở dữ liệu khi có sự thay đổi sẽ khó khăn.

Hướng nghiên cứu tiếp theo:

Qua những kết quả đã và chưa đạt được, em nhận thấy rằng khả năng và kiến thức còn nhiều hạn chế, để ứng dụng được hoàn thiện hơn, khắc phục được các mặt hạn chế.

Trong tương lai, cần tìm hiểu sâu hơn về ngôn ngữ Objective C, công cụ Xcode để giải quyết những vấn đề thường hay phát sinh ra lỗi như: tràn bộ nhớ,

không hiển thị hình ảnh, ... Ngoài ra còn cần tối ưu hoá ứng dụng để ứng dụng hoạt động một cách mượt mà và ổn định hơn tránh tình trạng bị lag, ...

Đối với các câu hỏi cũng cần làm cho các đáp án cũng được sắp xếp ngẫu nhiên để tránh trường hợp người dùng nhớ vị trí đáp án mà không nhớ nội dung.

TÀI LIỆU THAM KHẢO

TiếngViệt

[1]. Bộ Giao Thông Vận Tải (2012), Thông Tư ban hành "Quy chuẩn kỹ thuật quốc gia về báo hiệu đường bộ".

[2]. Bộ Giao Thông Vận Tải (2013), Hỏi và đáp về luật giao thông đường bộ.

[3]. Cao Thanh Vàng, Nguyễn Anh Tiệp (2013), Tài liệu hướng dẫn xây dựng ứng dụng iPhone.

[4]. GV. Trần Xuân Hoà Thắng (2013), Tóm tắt lý thuyết – mẹo làm bài thi sát hạch 2013.

[5]. Trung tâm Nhất Nghệ (2013), Lập trình ứng dụng iPhone cơ bản.

[6]. Trung tâm tin học đại học Khoa Học Tự Nhiên Tp Hồ Chí Minh (2014), Lập trình iOS.

Tiếng Anh

[7]. Apple Inc (2012), Your First iOS App.

[8]. David Mark, Jack Nutting, Jeff LaMarche, Fredrik Olsson (2013), Begiing iOS6 Development.

[9]. Neil Smyth (2013), iOS7 App Development Essentials.

Internet

[10]. Cấu trúc đề thi lý thuyết bằng lái xe, <http://daotaobanglaixe.net/cau-truc-de-thi-ly-thuyet-bang-lai-xe-oto-hang-b1-b2-c-d-post231.html>

[11]. Giải đáp ý nghĩa của các kiểu vạch kẻ đường, <http://duongbo.vn/2303-23710/Giai-dap-y-nghia-cua-cac-kieu-vach-ke-duong>

[12]. Hướng dẫn JailCoder, <https://www.youtube.com/watch?v=tXbGMFPL9lA>

[13]. How To Submit An App To Apple's App Store, <https://www.youtube.com/watch?v=rRlOdp4uZoo&app=desktop>

[14]. Những điểm mới ở bộ 450 câu hỏi lý thuyết lái xe, <https://www.facebook.com/xethegioi/posts/630285680323819>

[15]. Làm thế nào để đưa ứng dụng lên iPhone, <http://macvn.com/forums/showthread.php?t=37835>

[16]. Lưu trữ trong ứng dụng iOS OSx, <http://xcodeblog.com/2013/11/20/luu-tru-trong-ung-dung-iososx-nscoding-nskeyedarchiver.iosx>

[17]. Lý thuyết học lái xe, <http://banglaioto.com.vn/thi-ly-thuyet-lai-xe/ly-thuyet-hoc-lai-xe-o-to-b2-tai-tphcm_4.html>

[18]. Tình hình an toàn giao thông trong 9 tháng đầu năm 2014, <http://haiphong.gov.vn/Portal/Detail.aspx?Organization=ubndtp&MenuID=170&ContentID=62363>
<http://baobinhduong.vn/tinh-hinh-giao-thong-9-thang-dau-nam-2014-giam-ca-3-tieu-chi-a101200.html>