

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG**



ISO 9001:2015

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên : Bùi Trần Lĩnh

Giảng viên hướng dẫn: ThS. Nguyễn Trịnh Đông

HẢI PHÒNG - 2018

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG**

**KIỂM THỬ PHẦN MỀM TRÊN THIẾT BỊ DI ĐỘNG VÀ ỨNG
DỤNG PHẦN MỀM APPIUM STUDIO CHO ỨNG DỤNG
TRÊN IOS**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
NGÀNH: CÔNG NGHỆ THÔNG TIN**

Sinh viên : Bùi Trần Lĩnh

Giảng viên hướng dẫn : ThS. Nguyễn Trịnh Đông

HẢI PHÒNG - 2018

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG**

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Sinh viên: Bùi Trần Linh

Mã SV: 1412101135

Lớp: CT1801

Ngành: Công nghệ thông tin

Tên đề tài: Kiểm thử phần mềm trên thiết bị di động và ứng dụng phần mềm Appium Studio cho ứng dụng trên IOS

LỜI CẢM ƠN

Được sự phân công của Khoa Công nghệ thông tin Trường Đại Học Dân lập Hải Phòng, và dưới sự hướng dẫn của Thầy giáo hướng dẫn ThS. Nguyễn Trịnh Đông, em đã hoàn thành đề tài “Kiểm thử phần mềm trên thiết bị di động và ứng dụng phần mềm Appium Studio cho ứng dụng trên IOS”.

Để hoàn thành khóa luận này, em xin chân thành cảm ơn tới các thầy cô giáo đã tận tình hướng dẫn, giảng dạy trong suốt quá trình học tập, nghiên cứu và rèn luyện ở Trường Đại Học Dân lập Hải Phòng. Đặc biệt xin gửi lời cảm ơn chân thành tới Thầy giáo hướng dẫn ThS. Nguyễn Trịnh Đông đã tận tình, chu đáo hướng dẫn em thực hiện khoá luận này.

Mặc dù đã có nhiều cố gắng để thực hiện đề tài một cách hoàn chỉnh nhất. Song do thời gian có hạn, trình độ hiểu biết và nhận thức còn chưa cao cho nên trong đồ án không thể tránh khỏi những thiếu sót, em rất mong nhận được sự đóng góp ý kiến của các thầy cô và bạn bè để em có thể hoàn thiện đồ án này tốt hơn.

Em xin chân thành cảm ơn!

Hải Phòng, ngày 31 tháng 3 năm 2018

Sinh viên thực hiện

Bùi Trần Lĩnh

MỤC LỤC

LỜI CẢM ƠN.....	1
MỤC LỤC	5
DANH MỤC HÌNH VẼ VÀ BẢNG BIỂU	7
DANH MỤC TỪ VIẾT TẮT VÀ THUẬT NGỮ	8
MỞ ĐẦU.....	10
CHƯƠNG 1: CÁC KIẾN THỨC CƠ BẢN.....	13
1. Phần mềm	13
2. Kiểm thử phần mềm và một số khái niệm liên quan	13
2.1. Kiểm thử phần mềm	13
2.2. Một số khái niệm liên quan.....	14
3. Quy trình kiểm thử phần mềm	16
4. Các cấp độ kiểm thử	17
4.1. Kiểm thử mức đơn vị	18
4.2. Kiểm thử tích hợp	19
4.3. Kiểm thử hồi quy	19
4.4. Kiểm thử chấp nhận sản phẩm	20
4.5. Kiểm thử mức hệ thống.....	20
5. Các kỹ thuật kiểm thử phần mềm	20
5.1. Nguyên tắc cơ bản kiểm thử phần mềm	21
5.2. Kỹ thuật kiểm thử hộp trắng (White-Box Testing)	23
5.3. Kỹ thuật kiểm thử hộp đen (Black-Box Testing).....	25
6. Kỹ thuật thiết kế Ca kiểm thử	26
6.1. Cấu trúc của Ca kiểm thử	27
6.2. Phân vùng tương đương	28
6.3. Phân tích giá trị biên	31
6.4. Đoán lỗi.....	33
7. Tạo Bug report	34
7.1. Bug và Bug report.....	34
7.2. Cấu trúc một Bug report	34
7.3. Severity và Priority.....	36
CHƯƠNG 2: KIỂM THỬ TRÊN THIẾT BỊ DI ĐỘNG	38
1. Kiểm thử trên thiết bị di động	38
1.1. Các khái niệm cơ bản về ứng dụng di động.....	38

1.2. Phương pháp kiểm thử trên thiết bị di động	41
1.3. Các loại kiểm thử di động	45
1.4. Các đặc điểm của kiểm thử di động	46
2. Kiểm thử tự động	48
2.1. Khái niệm kiểm thử tự động	48
2.2. Mục tiêu của kiểm thử tự động	48
2.3. Nguyên tắc kiểm thử tự động	50
2.4. Quy trình kiểm thử tự động	52
2.5. Ưu điểm của kiểm thử tự động	53
2.6. Một số công cụ kiểm thử tự động	54
2.7. So sánh kiểm thử tự động và kiểm thử thủ công	55
CHƯƠNG 3: THỰC NGHIỆM SỬ DỤNG APPIUM STUDIO CHO KIỂM THỬ TỰ ĐỘNG TRÊN IOS	57
1. Giới thiệu phần mềm Appium Studio	57
1.1. Công cụ Appium	57
1.2. Phần mềm Appium Studio	58
2. Appium Studio tích hợp trong Eclipse	59
3. Thực nghiệm với Appium Studio tích hợp trong Eclipse	60
3.1. Cài đặt Appium Studio	60
3.2. Kết nối với thiết bị trên Cloud	61
3.3. Xây dựng bộ ca kiểm thử cho một ứng dụng cần kiểm thử	63
3.4. Tạo dự án kiểm thử	65
3.5. Báo cáo	74
KẾT LUẬN	78
TÀI LIỆU THAM KHẢO	79

DANH MỤC HÌNH VẼ VÀ BẢNG BIỂU

Hình 1-1: Ví dụ về 1 Kịch bản kiểm thử	16
Hình 1-2: Giai đoạn kiểm thử trong xử lý phần mềm.....	16
Hình 1-3: Luồng thông tin kiểm thử	22
Hình 1-4: Minh họa Kiểm thử hộp đen.....	26
Hình 1-5: Minh họa của một ca kiểm thử	28
Hình 1-6: Minh họa một Form đăng nhập	29
Hình 1-7: Minh họa một Bug report	35
Hình 2-1: Quy trình Kiểm thử tự động trong mối quan hệ với Kiểm thử phần mềm.....	53
Bảng 2-2: So sánh kiểm thử tự động và kiểm thử thủ công.....	56
Hình 3-1: Kết quả tìm kiếm Appium Studio.....	60
Hình 3-2: Lấy URL để cài đặt Appium Studio	60
Hình 3-3: Dán URL vào cửa sổ Install để tiến hành cài đặt	61
Hình 3-4: Giao diện trang Cloud của SeeTest	61
Hình 3-5: Copy lại Access Key	62
Hình 3-6: Kiểm tra kết nối đến máy chủ Cloud.....	62
Hình 3-7: Các thiết bị Cloud được hiển thị trong Eclipse	62
Hình 3-8: Màn hình thiết bị được hiển thị sau khi kết nối.....	63
Hình 3-9: Giao diện chương trình máy tính cần kiểm thử.....	63
Hình 3-10: Bộ ca kiểm thử cho ứng dụng máy tính	65
Hình 3-11: Đoạn code IOSTest được sinh tự động trong Project.....	66
Hình 3-12: Kết quả tìm kiếm “TestNG”	66
Hình 3-13: Kết quả sau khi cài đặt ứng dụng Basic Calculator.....	67
Hình 3-14: Code cài đặt ứng dụng được thêm vào phần setUp.....	68
Hình 3-15: Thêm câu lệnh để chương trình không tự động thoát khi thực hiện kiểm thử ..	68
Hình 3-16: Chọn biểu tượng Dump UI ở cửa sổ Devices.....	69
Hình 3-17: Màn hình được lưu với tên “mainScreen.dump”	69
Hình 3-18: Lưu lại đối tượng nút AC của màn hình máy tính.....	70
Hình 3-19: Đoạn mã sinh số thập phân ngẫu nhiên từ -999 đến 999	71
Hình 3-20: Đoạn mã sinh số nguyên ngẫu nhiên từ -999 đến 999	71
Hình 3-21: Đoạn mã sinh dữ liệu kiểm thử tự động	73
Hình 3-22: Khởi chạy kiểm thử tự động.....	73
Hình 3-23: Quá trình chạy kiểm thử trên web	74
Hình 3-24: Kết quả sinh ca kiểm thử tự động.....	74
Hình 3-25: Toàn bộ báo cáo được sinh tự động trong phần Reports.....	75
Hình 3-26: Chi tiết quá trình thực hiện kiểm thử tự động	75
Hình 3-27: Ca kiểm thử đầu tiên không đưa ra kết quả chính xác	75
Hình 3-28: Tổng hợp kết quả kiểm thử.....	76
Hình 3-29: Bug report lỗi của nút (+/-).....	77

DANH MỤC TỪ VIẾT TẮT VÀ THUẬT NGỮ

STT	KÝ HIỆU	CỤM TỪ ĐẦY ĐỦ	Ý NGHĨA
1	3G	Third-generation technology	Công nghệ truyền thông thế hệ thứ ba, cho phép truyền cả dữ liệu thoại và dữ liệu ngoài thoại
2	API	Application Programming Interface	Giao diện lập trình ứng dụng - là 1 giao tiếp phần mềm được dùng bởi các ứng dụng khác nhau
3	BSD	Berkeley Software Distribution	Tên của một hệ điều hành dẫn xuất từ UNIX được phát hành vào thập niên 1970 từ trường Đại học California tại Berkeley
4	CPU	Central Processing Unit	Bộ xử lý trung tâm
5	Framework	Framework	Framework là một thư viện các lớp đã được xây dựng hoàn chỉnh, bộ khung để phát triển các Phần mềm ứng dụng
6	GPRS	General Packet Radio Service	Dịch vụ vô tuyến gói tổng hợp - là một dịch vụ dữ liệu di động dạng gói dành cho những người dùng Hệ thống thông tin di động toàn cầu
7	GPS	Global Positioning System	Hệ thống định vị toàn cầu
8	GSM	Global System for Mobile Communications	Một công nghệ dùng cho mạng thông tin di động
9	HTTP	HyperText Transfer Protocol	Giao thức truyền tải siêu văn bản
10	ID	Identification number	Mã số

11	IDE	Integrated Development Environment	Phần mềm bao gồm những gói phần mềm khác giúp phát triển ứng dụng phần mềm (Môi trường phát triển tích hợp)
12	IEEE	Institute of Electrical and Electronics Engineers	Viện kỹ nghệ Điện và Điện tử
13	IT	Information Technology	Công nghệ thông tin
14	Linux	Linux	Tên gọi của một hệ điều hành máy tính và cũng là tên hạt nhân của hệ điều hành.
15	QA	Quality Assurance	Người chịu trách nhiệm đảm bảo chất lượng sản phẩm
16	SDK	Software Development Kit	Thuật ngữ được Microsoft, Sun Microsystems và một số công ty khác sử dụng – một bộ công cụ phát triển phần mềm
17	SMS	Short Message Services	Giao thức viễn thông cho phép gửi các thông điệp dạng text ngắn
18	SQA	Software Quality Assurance	Tập hợp các hoạt động đảm bảo chất lượng trong quá trình gia công phần mềm
19	UI	User Interface	Giao diện người dùng
20	URL	Uniform Resource Locator	Định vị tài nguyên thống nhất, được dùng để tham chiếu tới tài nguyên trên Internet
21	V&V	Verification and Validation	Xác minh và thẩm định
22	WAP	Wireless Application Protocol	Giao thức Ứng dụng không dây - là một tiêu chuẩn công nghệ cho các hệ thống truy nhập Internet từ các thiết bị di động

MỞ ĐẦU

Lý do chọn đề tài:

Với sự phát triển như vũ bão của công nghệ thông tin nói chung và công nghệ phần mềm nói riêng, việc phát triển phần mềm ngày càng được hỗ trợ bởi nhiều công cụ tiên tiến, giúp cho việc xây dựng phần mềm đỡ mệt nhọc và hiệu quả hơn. Tuy nhiên, vì độ phức tạp của phần mềm và những giới hạn về thời gian và chi phí, cho dù các hoạt động đảm bảo chất lượng phần mềm nói chung và kiểm thử nói riêng ngày càng chặt chẽ và khoa học, vẫn không đảm bảo được rằng các sản phẩm phần mềm đang được ứng dụng không có lỗi. Lỗi vẫn luôn tiềm ẩn trong mọi sản phẩm phần mềm và cũng có thể gây những thiệt hại khôn lường.

Kiểm thử phần mềm là một quá trình liên tục, xuyên suốt mọi giai đoạn phát triển phần mềm để đảm bảo rằng phần mềm thoả mãn các yêu cầu thiết kế và các yêu cầu đó đáp ứng các nhu cầu của người dùng. Các kỹ thuật kiểm thử phần mềm đã và đang được nghiên cứu, và việc kiểm thử phần mềm đã trở thành quy trình bắt buộc trong các dự án phát triển phần mềm trên thế giới. Kiểm thử phần mềm là một hoạt động rất tốn kém, mất thời gian, và khó phát hiện được hết lỗi. Vì vậy, việc kiểm thử phần mềm đòi hỏi phải có chiến lược phù hợp, một kế hoạch hợp lý và việc thực hiện được quản lý chặt chẽ.

Và với việc những chiếc điện thoại thông minh đang ngày càng được sử dụng nhiều hơn nhằm đáp ứng nhu cầu giải trí đa dạng của người dùng. Từ một chiếc điện thoại thông thường chỉ được cài đặt sẵn vài ba ứng dụng của nhà sản xuất thì nay với các thiết bị chạy các hệ điều hành nhúng (Android, iOS, v.v.) ta có thể dễ dàng đáp ứng được các nhu cầu của người dùng bằng cách cài thêm các phần mềm bên thứ ba mà không gây ra trở ngại nào. Từ đây lại đặt ra một vấn đề hiển nhiên là kiểm thử các phần mềm chạy trên di động này để xem chúng có đáp ứng được các yêu cầu đề ra ban đầu hay không trước khi phát hành sản phẩm tới tay người tiêu dùng.

Đó là lý em chọn đề tài “Kiểm thử phần mềm trên thiết bị di động và ứng dụng phần mềm Appium Studio cho ứng dụng trên IOS” làm đồ án tốt nghiệp.

Mục đích của đồ án:

Đề tài tìm hiểu cơ sở lý thuyết về kiểm thử nói chung và kiểm thử trên di động nói riêng cũng như cách triển khai công cụ kiểm thử phần mềm tự động để giảm nhân lực kiểm thử và đảm bảo chất lượng phần mềm hơn với công việc kiểm thử bằng tay. Mục tiêu chính của đề tài là nghiên cứu về kiểm thử trên thiết bị di động.

Đối tượng và phạm vi nghiên cứu:

Đồ án nghiên cứu lý thuyết kiểm thử phần mềm. Bên cạnh đó, nghiên cứu các vấn đề về kiểm thử phần mềm trên thiết bị di động và ứng dụng phần mềm Appium Studio cho kiểm thử tự động trên IOS.

Phương pháp nghiên cứu:

Nghiên cứu tổng quan về kiểm thử phần mềm và các kỹ thuật kiểm thử từ đó áp dụng vào kiểm thử phần mềm trên thiết bị di động, tìm hiểu công cụ kiểm thử phần mềm Appium Studio trên IOS.

Với mục tiêu đặt ra như vậy, những nội dung và kết quả nghiên cứu chính của đồ án được trình bày trong ba chương như sau:

Chương 1: Các kiến thức cơ bản

Chương 2: Kiểm thử trên thiết bị di động

Chương 3: Thực nghiệm sử dụng phần mềm Appium Studio cho kiểm thử tự động trên IOS

Phần kết luận đưa ra những đánh giá về những kết quả đạt được và những khó khăn gặp phải trong quá trình nghiên cứu thực hiện đồ án.

Trong quá trình thực hiện đồ án, do thời gian cũng như trình độ của em còn có những hạn chế nhất định nên không thể tránh khỏi những sai sót. Rất mong nhận được sự góp ý của các thầy, cô giáo và các bạn để đồ án hoàn thiện hơn. Em xin chân thành cảm ơn sự hướng dẫn, và giúp đỡ tận tình của thầy giáo **ThS. Nguyễn Trịnh Đông**, các thầy cô trong khoa Công nghệ thông tin Trường Đại học Dân lập Hải Phòng đã giúp đỡ em trong quá trình học tập cũng như trong quá trình làm đồ án.

CHƯƠNG 1: CÁC KIẾN THỨC CƠ BẢN

Kiểm thử nhằm đánh giá chất lượng hoặc tính chấp nhận được của sản phẩm. Ngoài ra, kiểm thử còn giúp phát hiện lỗi hoặc bất cứ vấn đề gì về sản phẩm. Chúng ta cần kiểm thử vì biết rằng con người luôn có thể mắc sai lầm. Điều này đặc biệt đúng trong lĩnh vực phát triển phần mềm và các hệ thống điều khiển bởi phần mềm. Chương này sẽ giới thiệu các khái niệm trong lĩnh vực kiểm thử phần mềm.

1. Phần mềm

Phần mềm thường được mô tả bởi ba thành phần cấu thành [1]:

- *Tập các lệnh* (chương trình máy tính) trên máy tính khi thực hiện sẽ tạo ra các dịch vụ và đem lại những kết quả mong muốn cho người dùng.
- *Các cấu trúc dữ liệu* (lưu giữ trên các bộ nhớ) làm cho chương trình thao tác hiệu quả với các thông tin thích hợp và nội dung thông tin được số hóa.
- *Các tài liệu* để mô tả thao tác, cách sử dụng và bảo trì phần mềm (hướng dẫn sử dụng, tài liệu kỹ thuật, tài liệu phân tích, thiết kế, kiểm thử, v.v.).

2. Kiểm thử phần mềm và một số khái niệm liên quan

2.1. Kiểm thử phần mềm

Kiểm thử phần mềm là một cuộc kiểm tra được tiến hành để cung cấp cho các bên liên quan thông tin về chất lượng của sản phẩm hoặc dịch vụ được kiểm thử [2]. Kiểm thử có thể cung cấp cho doanh nghiệp một quan điểm, một cách nhìn độc lập về phần mềm để từ đó cho phép đánh giá và thấu hiểu được những rủi ro trong quá trình triển khai phần mềm.

Trong kỹ thuật kiểm thử không chỉ giới hạn ở việc thực hiện một chương trình hoặc ứng dụng với mục đích đi tìm các lỗi phần mềm (bao gồm các lỗi và

các thiếu sót) mà còn là một quá trình phê chuẩn và xác minh một chương trình máy tính / ứng dụng / sản phẩm nhằm:

Đáp ứng được mọi yêu cầu hướng dẫn khi thiết kế và phát triển phần mềm.

Thực hiện công việc đúng như kỳ vọng.

Có thể triển khai được với những đặc tính tương tự.

Và đáp ứng được mọi nhu cầu của các bên liên quan.

Tùy thuộc vào từng phương pháp, việc kiểm thử có thể được thực hiện bất cứ lúc nào trong quá trình phát triển phần mềm. Theo truyền thống thì các nỗ lực kiểm thử được tiến hành sau khi các yêu cầu được xác định và việc lập trình được hoàn tất nhưng trong Agile (là một tập hợp các phương pháp phát triển phần mềm linh hoạt dựa trên việc lặp đi lặp lại và gia tăng giá trị) thì việc kiểm thử được tiến hành liên tục trong suốt quá trình xây dựng phần mềm. Như vậy, mỗi một phương pháp kiểm thử bị chi phối theo một quy trình phát triển phần mềm nhất định.

2.2. Một số khái niệm liên quan

Chất lượng phần mềm (Software quality): là mức độ mà một hệ thống, thành phần hay quy trình đáp ứng các yêu cầu của đặc tả phần mềm, các nhu cầu mong đợi của khách hàng hoặc người sử dụng [3].

Đảm bảo chất lượng phần mềm (Software quality assurance): là một quy trình có kế hoạch và hệ thống của tất cả các hành động cần thiết để cung cấp các thông tin đầy đủ để đảm bảo các sản phẩm có phù hợp với các yêu cầu về kỹ thuật hay không. Mục đích cuối cùng là để đánh giá quy trình sản xuất sản phẩm phần mềm [3].

Xác nhận (Validation): là quá trình đánh giá một hệ thống hay cấu phần trong hay cuối của quá trình phát triển để xác định xem nó đáp ứng yêu cầu quy định [3].

Xác minh, kiểm chứng (Verification): là quá trình đánh giá một hệ thống hay thành phần để xác định xem các sản phẩm của một giai đoạn phát triển nhất định đáp ứng các điều kiện áp đặt tại lúc bắt đầu của giai đoạn đó [3]. Xác minh thường là hoạt động có tính kỹ thuật cao hơn, sử dụng những tri thức về các yêu cầu, đặc tả phần mềm. Xác nhận thường phụ thuộc vào tri thức về lĩnh vực tương ứng. Cụ thể là, tri thức về ứng dụng của phần mềm được viết. Ví dụ, xác nhận của phần mềm về máy bay yêu cầu tri thức từ kỹ sư hàng không và phi công.

Lỗi (Error): Lỗi là những vấn đề mà con người mắc phải trong quá trình phát triển các sản phẩm phần mềm [4].

Sai (Fault): Sai là kết quả của lỗi, hay nói khác đi, lỗi sẽ dẫn đến sai [4].

Thất bại (Failure): Thất bại xuất hiện khi một lỗi được thực thi [4].

Sự cố (Incident): Khi thất bại xuất hiện, nó có thể hiển thị hoặc không, tức là rõ ràng hoặc không rõ ràng đối với người dùng hoặc người kiểm thử. Sự cố là triệu chứng liên kết với một thất bại và thể hiện cho người dùng hoặc người kiểm thử về sự xuất hiện của thất bại này [4].

Ca kiểm thử (Test case): Ca kiểm thử gồm một tập các dữ liệu đầu vào và một xâu các giá trị đầu ra mong đợi đối với phần mềm, mục đích là dựa vào đó để kiểm tra xem phần mềm có thỏa các yêu cầu đặt ra hay không.

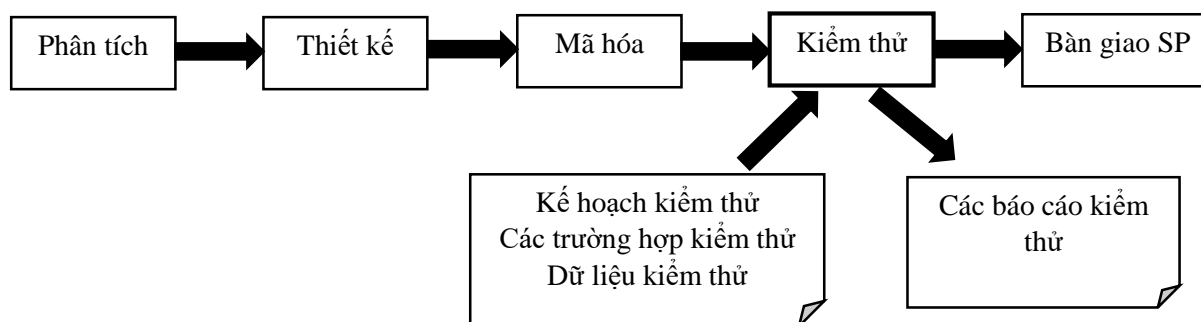
Kịch bản kiểm thử (Test script): Một kịch bản kiểm thử là một nhóm mã lệnh dạng đặc tả kịch bản dùng để tự động hóa một quy trình hay một ca kiểm tra, giúp cho việc kiểm tra nhanh hơn, hoặc cho những trường hợp mà kiểm tra bằng tay sẽ rất khó khăn hoặc không khả thi.

```
nRowCount=DataTable.GlobalSheet.GetRowCount
systemutil.CloseProcessByName "iexplore.exe"
For i = 1 to nRowCount
  Systemutil.Run "iexplore", "http://abc.com"
  Browser("title:=.*").Page("title:=.*").Sync
  DataTable.GlobalSheet.SetCurrentRow(i)
  sEmailid=DataTable("UserName",Global)
  sPassword=Datatable("Password",dtGlobalSheet)
  Browser("title:=.*").Page("title:=.*").webEdit("name=Email").Set sEmailid
  Browser("title:=.*").Page("title:=.*").webEdit("name=Passwd").Set sPassword
  Browser("title:=.*").Page("title:=.*").webButton("name=Sign in").Click
  Browser("title:=.*").Page("title:=.*").Sync
  wait(10)
  If left(browser("title:=.*").Page("title:=.*").getROProperty("title"),13) ="welcome home"Then
    MsgBox("signed in successfully")
  Else
    MsgBox("Login Unsuccessful")
  End If
  Systemutil.CloseProcessByName "iexplore.exe"
Next
```

Hình 1-1: Ví dụ về 1 Kịch bản kiểm thử

3. Quy trình kiểm thử phần mềm

Mục đích của kiểm thử là thiết kế một chuỗi các trường hợp kiểm thử mà có khả năng phát hiện lỗi cao. Để cho việc kiểm thử đạt được kết quả tốt cần có sự chuẩn bị về kế hoạch kiểm thử, thiết kế các trường hợp kiểm thử và các dữ liệu kiểm thử cho các trường hợp. Đây chính là đầu vào cho giai đoạn kiểm thử. Và sản phẩm công việc của giai đoạn kiểm thử chính là “báo cáo kiểm thử” mà tài liệu hóa tất cả các trường hợp kiểm thử đã chạy, dữ liệu đầu vào, đầu ra mong đợi, đầu ra thực tế và mục đích của kiểm thử.



Hình 1-2: Giai đoạn kiểm thử trong xử lý phần mềm

Quy trình kiểm thử bao gồm một số giai đoạn:

- Lập kế hoạch kiểm thử: Bước đầu tiên là lập kế hoạch cho tất cả các hoạt động sẽ được thực hiện và các phương pháp được sử dụng. Các chuẩn IEEE bao gồm các thông tin về tác giả chuẩn bị kế hoạch, danh sách liệt kê của kế hoạch kiểm thử. Vấn đề quan trọng nhất đối với kế hoạch kiểm thử:

- Mục đích: Quy định về phạm vi, phương pháp, tài nguyên và lịch biểu của các hoạt động kiểm thử.
 - Các tài liệu tham khảo.
 - Các định nghĩa.
 - Khái quát về xác minh và thẩm định (V&V): tổ chức, tài nguyên, trách nhiệm, các công cụ, kỹ thuật và các phương pháp luận.
 - Vòng đời của V&V: các nhiệm vụ, các dữ liệu vào và các kết quả ra trên một giai đoạn vòng đời.
 - Báo cáo xác minh và thẩm định(V&V) phần mềm: mô tả nội dung, định dạng và thời gian cho tất cả các báo cáo V&V.
 - Các thủ tục quản lý V&V bao gồm các chính sách, thủ tục, các chuẩn, thực nghiệm và các quy ước.
- Giai đoạn bố trí nhân viên kiểm thử: Việc kiểm thử thường phải tiến hành một cách độc lập và các nhóm độc lập có trách nhiệm tiến hành các hoạt động kiểm thử, gọi là các nhóm kiểm thử.
 - Thiết kế các trường hợp kiểm thử: Các trường hợp kiểm thử là các đặc tả đầu vào cho kiểm thử và đầu ra mong đợi của hệ thống cùng với các câu lệnh được kiểm thử.
 - Các kỹ thuật kiểm thử hộp đen để kiểm thử dựa trên chức năng.
 - Các kỹ thuật kiểm thử hộp trắng để kiểm thử dựa vào cấu trúc bên trong.
 - Xử lý đo lường kiểm thử bằng cách thu thập dữ liệu.
 - Đánh giá sản phẩm phần mềm để xác nhận sản phẩm có thể sẵn sàng phát hành được chưa?

4. Các cấp độ kiểm thử

Các mức kiểm thử phần mềm thông thường:

- *Unit Test* – Kiểm thử mức đơn vị
- *Integration Test* – Kiểm thử tích hợp
- *System Test* - Kiểm thử mức hệ thống

- *Acceptance Test* - Kiểm thử chấp nhận sản phẩm
- *Regression Test* - Kiểm thử hồi quy

4.1. Kiểm thử mức đơn vị

Một đơn vị kiểm thử là một thành phần phần mềm nhỏ nhất mà ta có thể kiểm thử được. Theo định nghĩa này, các hàm (Function), thủ tục (Procedure), lớp (Class), hoặc các phương thức (Method) đều có thể được xem là đơn vị kiểm thử.

Vì đơn vị kiểm thử được chọn để kiểm thử thường có kích thước nhỏ và chức năng hoạt động đơn giản, chúng ta không khó khăn gì trong việc tổ chức, kiểm thử, ghi nhận và phân tích kết quả kiểm thử. Nếu phát hiện lỗi, việc xác định nguyên nhân và khắc phục cũng tương đối dễ dàng vì chỉ khoanh vùng trong một đơn vị đang kiểm thử. Một nguyên lý đúc kết từ thực tiễn: thời gian tốn cho Kiểm thử đơn vị sẽ được đền bù bằng việc tiết kiệm rất nhiều thời gian và chi phí cho việc kiểm thử và sửa lỗi ở các mức kiểm thử sau đó.

Kiểm thử đơn vị thường do lập trình viên thực hiện. Công đoạn này cần được thực hiện càng sớm càng tốt trong giai đoạn viết code và xuyên suốt chu kỳ phát triển phần mềm. Thông thường, Kiểm thử đơn vị đòi hỏi kiểm thử viên có kiến thức về thiết kế và mã nguồn của chương trình. Mục đích của Kiểm thử đơn vị là bảo đảm thông tin được xử lý và xuất ra là chính xác, trong mối tương quan với dữ liệu nhập và chức năng của đơn vị kiểm thử. Điều này thường đòi hỏi tất cả các nhánh bên trong đơn vị kiểm thử đều phải được kiểm tra để phát hiện nhánh phát sinh lỗi. Một nhánh thường là một chuỗi các lệnh được thực thi trong một đơn vị kiểm thử, ví dụ: chuỗi các lệnh sau điều kiện If và nằm giữa then ... else là một nhánh. Thực tế việc chọn lựa các nhánh để đơn giản hóa việc kiểm thử và quét hết các đơn vị kiểm thử đòi hỏi phải có kỹ thuật, đôi khi phải dùng thuật toán để chọn lựa.

Cũng như các mức kiểm thử khác, Kiểm thử đơn vị cũng đòi hỏi phải chuẩn bị trước các ca kiểm thử hoặc kịch bản kiểm thử, trong đó chỉ định rõ dữ

liệu vào, các bước thực hiện và dữ liệu mong chờ sẽ xuất ra. Các ca kiểm thử và kịch bản này nên được giữ lại để tái sử dụng.

Kiểm thử đơn vị thường sử dụng các Unit Test Framework, đó là các khung chương trình được viết sẵn để hỗ trợ cho việc test các mô đun, các đơn vị phần mềm.

4.2. Kiểm thử tích hợp

Kiểm thử tích hợp kết hợp các thành phần của một ứng dụng và kiểm thử như một ứng dụng đã hoàn thành. Trong khi Kiểm thử đơn vị kiểm thử các thành phần và đơn vị phần mềm riêng lẻ thì kiểm thử tích hợp kết hợp chúng lại với nhau và kiểm thử sự giao tiếp giữa chúng. Kiểm thử tích hợp có 2 mục tiêu chính:

- Phát hiện lỗi giao tiếp xảy ra giữa các đơn vị kiểm thử.
- Tích hợp các đơn vị kiểm thử đơn lẻ thành các hệ thống nhỏ (subsystem) và cuối cùng là nguyên hệ thống hoàn chỉnh (system) chuẩn bị cho kiểm thử ở mức hệ thống.

4.3. Kiểm thử hồi quy

Kiểm thử hồi quy không phải là một mức kiểm thử, như các mức khác đã nói ở trên. Nó đơn thuần kiểm tra lại phần mềm sau khi có một sự thay đổi xảy ra, để bảo đảm phiên bản phần mềm mới thực hiện tốt các chức năng như phiên bản cũ và sự thay đổi không gây ra lỗi mới trên những chức năng vốn đã làm việc tốt. Kiểm thử hồi quy có thể thực hiện tại mọi mức kiểm thử. Ví dụ: một phần mềm đang phát triển khi kiểm tra cho thấy nó chạy tốt các chức năng A, B và C. Khi có thay đổi code của chức năng C, nếu chỉ kiểm tra chức năng C thì chưa đủ, cần phải kiểm tra lại tất cả các chức năng khác liên quan đến chức năng C, trong ví dụ này là A và B. Lý do là khi C thay đổi, nó có thể sẽ làm A và B không còn làm việc đúng nữa.

4.4. Kiểm thử chấp nhận sản phẩm

Thông thường, sau giai đoạn kiểm thử hệ thống là kiểm thử chấp nhận, được khách hàng thực hiện (hoặc ủy quyền cho một nhóm thứ ba thực hiện). Mục đích của kiểm thử chấp nhận là để chứng minh phần mềm thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm (và trả tiền thanh toán hợp đồng). Kiểm thử chấp nhận có ý nghĩa hết sức quan trọng, mặc dù trong hầu hết mọi trường hợp, các phép kiểm thử của kiểm thử hệ thống và kiểm thử chấp nhận gần như tương tự, nhưng bản chất và cách thức thực hiện lại rất khác biệt.

4.5. Kiểm thử mức hệ thống

Mục đích Kiểm thử mức hệ thống là kiểm tra thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không. Điểm khác nhau then chốt giữa kiểm thử tích hợp và kiểm thử hệ thống là kiểm thử hệ thống chú trọng các hành vi và lỗi trên toàn hệ thống, còn kiểm thử tích hợp chú trọng sự giao tiếp giữa các đơn vị hoặc đối tượng khi chúng làm việc cùng nhau. Thông thường ta phải thực hiện kiểm thử đơn vị và kiểm thử tích hợp để bảo đảm mọi đơn vị phần mềm và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện kiểm thử hệ thống. Kiểm thử hệ thống kiểm tra cả các hành vi chức năng của phần mềm lẫn các yêu cầu về chất lượng như độ tin cậy, tính tiện lợi khi sử dụng, hiệu năng và bảo mật. Mức kiểm thử này đặc biệt thích hợp cho việc phát hiện lỗi giao tiếp với phần mềm hoặc phần cứng bên ngoài, chẳng hạn các lỗi “bế tắc” (deadlock) hoặc chiếm dụng bộ nhớ. Sau giai đoạn kiểm thử hệ thống, phần mềm thường đã sẵn sàng cho khách hàng hoặc người dùng cuối cùng kiểm thử để chấp nhận hoặc dùng thử (Alpha/Beta Test).

5. Các kỹ thuật kiểm thử phần mềm

Có thể chia các kỹ thuật kiểm thử phần mềm thành hai loại: các kỹ thuật kiểm thử hộp đen (black-box testing) và kỹ thuật kiểm thử hộp trắng (white-box testing). Các kiểm thử hộp đen tìm các lỗi như thiếu các chức năng, khả

năng sử dụng và các yêu cầu phi chức năng. Trong khi các kỹ thuật kiểm thử hộp trắng yêu cầu hiểu biết về cấu trúc chương trình bên trong và các kiểm thử nhận được từ đặc tả thiết kế bên trong hoặc từ mã.

5.1. Nguyên tắc cơ bản kiểm thử phần mềm

Trong lúc kiểm thử, công nghệ phần mềm phát sinh một chuỗi các trường hợp kiểm thử được sử dụng để “tách từng phần” phần mềm. Kiểm thử là một bước trong quy trình phần mềm mà có thể được xem xét bởi đội ngũ phát triển bằng cách phá vỡ thay vì xây dựng. Các kỹ sư phần mềm chính là những người xây dựng và kiểm thử yêu cầu họ vượt qua các khái niệm cho trước về độ chính xác và giải quyết mâu thuẫn khi các lỗi được xác định.

5.1.1. Mục tiêu kiểm thử

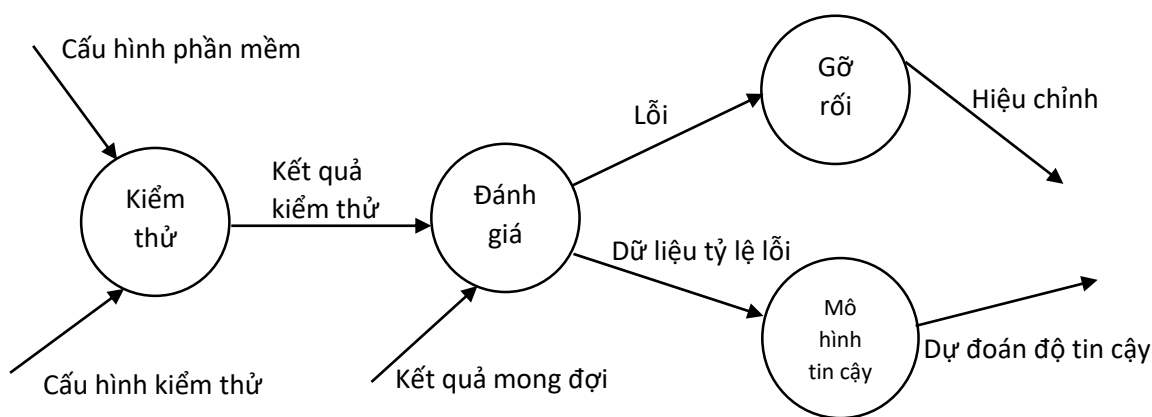
Các nguyên tắc được xem như mục tiêu kiểm thử là:

- Kiểm thử là một quá trình thực thi chương trình với mục đích tìm lỗi.
- Một trường hợp kiểm thử tốt là trường hợp kiểm thử mà có khả năng cao việc tìm thấy các lỗi chưa từng được phát hiện.
- Một kiểm thử thành công là kiểm thử mà phát hiện lỗi chưa từng được phát hiện.

5.1.2. Luồng thông tin kiểm thử

Luồng thông tin cho kiểm thử được biểu diễn bởi mô hình trong Hình 1.4. Hai kiểu của đầu vào được truyền cho quá trình kiểm thử:

- Cấu hình phần mềm: gồm các đặc tả yêu cầu, đặc tả thiết kế, và mã nguồn.
- Cấu hình kiểm thử: gồm có kế hoạch kiểm thử, các thủ tục, trường hợp kiểm thử, và các công cụ kiểm thử.



Hình 1-3: Luồng thông tin kiểm thử

5.1.3. Thiết kế trường hợp kiểm thử

Thiết kế kiểm thử phần mềm có thể là một quá trình thu thập, phân tích và thực hiện yêu cầu. Mục tiêu của kiểm thử là phải thiết kế các trường hợp kiểm thử có khả năng cao nhất trong việc phát hiện nhiều lỗi nhất với thời gian và công sức tối thiểu. Như vậy, vấn đề quan trọng nhất trong kiểm thử phần mềm là thiết kế và tạo ra các trường hợp kiểm thử có hiệu quả. Lý do về tầm quan trọng của việc thiết kế các trường hợp kiểm thử xuất phát từ thực tế: Kiểm thử “vét cạn” là điều không thể, và như vậy, kiểm thử một chương trình phải luôn xác định là không thể vét cạn. Vấn đề quan trọng là cố gắng làm giảm sự “không thể vét cạn” nhiều nhất có thể.

Kiểm thử phần mềm còn có các ràng buộc về thời gian, chi phí, v.v. Chìa khoá của kiểm thử là trả lời của câu hỏi: “Tập con của tất cả các trường hợp kiểm thử có thể có xác suất phát hiện lỗi cao nhất là gì?”. Việc nghiên cứu các phương pháp thiết kế trường hợp kiểm thử sẽ cung cấp câu trả lời cho câu hỏi này.

Bất kỳ sản phẩm công nghệ nào có thể được kiểm thử trong hai cách:

- Biết về các chức năng cụ thể mà sản phẩm đã được thiết kế để thực hiện.
- Biết cách hoạt động bên trong của sản phẩm, kiểm thử có thể được thực hiện để đảm bảo rằng “tất cả các thành phần ăn khớp nhau”.

Cách tiếp cận kiểm thử đầu tiên được gọi là kiểm thử hộp đen và cách thứ hai là kiểm thử hộp trắng.

5.2. Kỹ thuật kiểm thử hộp trắng (White-Box Testing)

Kiểm thử hộp trắng: Là kỹ thuật kiểm thử dựa trên đặc tả bên trong của chương trình, dựa vào mã nguồn, cấu trúc chương trình. Kiểm thử hộp trắng thường phát hiện các lỗi lập trình. Loại kiểm thử này khá khó thực hiện và chi phí cao.

Với các module quan trọng, thực thi việc tính toán chính của hệ thống, phương pháp này là cần thiết.

Có 2 kỹ thuật kiểm thử hộp trắng phổ biến:

5.2.1. Kiểm thử luồng dữ liệu

Phương pháp kiểm thử luồng dữ liệu lựa chọn các đường dẫn kiểm thử của chương trình dựa vào vị trí khai báo và sử dụng các biến trong chương trình. Với kiểm thử luồng dữ liệu mỗi câu lệnh trong chương trình được gán số hiệu lệnh duy nhất và mỗi hàm không thay đổi tham số của nó và biến toàn cục. Cho một lệnh với S là số hiệu câu lệnh. Ta định nghĩa,

$DEF(S)$ = là tập các biến được khai báo trong S .

$USE(S)$ = là tập các biến được sử dụng trong S .

Một chiến lược kiểm thử luồng dữ liệu cơ bản là chiến lược mà mỗi chuỗi DU được phủ ít nhất một lần. Chiến lược này được gọi là chiến lược kiểm thử DU. Kiểm thử DU không đảm bảo phủ hết tất cả các nhánh của một chương trình. Tuy nhiên, một nhánh không đảm bảo được phủ bởi kiểm thử DU chỉ trong rất ít tình huống như cấu trúc *if-then-else* mà trong đó phần *then* không có một khai báo biến nào và có dạng khuyết (không tồn tại phần *else*). Trong tình huống đó, nhánh *else* của lệnh *if* là không cần thiết phải phủ bằng kiểm thử DU.

Chiến lược kiểm thử luồng dữ liệu là rất hữu ích cho việc lựa chọn các đường dẫn kiểm thử của chương trình có chứa các lệnh *if* hoặc vòng lặp lồng nhau.

5.2.2. Kiểm thử luồng điều khiển

Đường thi hành (Execution path): là 1 kịch bản thi hành đơn vị phần mềm tương ứng: danh sách có thứ tự các lệnh được thi hành ứng với 1 lần chạy cụ thể của đơn vị phần mềm, bắt đầu từ điểm nhập của đơn vị phần mềm đến điểm kết thúc của đơn vị phần mềm.

Mục tiêu của phương pháp kiểm thử luồng điều khiển là đảm bảo mọi đường thi hành của đơn vị phần mềm cần kiểm thử đều chạy đúng. Rất tiếc trong thực tế, công sức và thời gian để đạt mục tiêu trên đây là rất lớn, ngay cả trên những đơn vị phần mềm nhỏ. Mà cho dù có kiểm thử hết được toàn bộ các đường thi hành thì vẫn không thể phát hiện những đường thi hành cần có nhưng không (chưa) được hiện thực. Do đó, ta nên kiểm thử số ca kiểm thử tối thiểu mà kết quả độ tin cậy tối đa.

Phủ kiểm thử (Coverage): là tỉ lệ các thành phần thực sự được kiểm thử so với tổng thể sau khi đã kiểm thử các ca kiểm thử được chọn. Phủ càng lớn thì độ tin cậy càng cao. Thành phần liên quan có thể là lệnh, điểm quyết định, điều kiện con, đường thi hành hay là sự kết hợp của chúng.

- *Phủ cấp 0*: kiểm thử những gì có thể kiểm thử được, phần còn lại để người dùng phát hiện và báo lại sau. Đây là mức độ kiểm thử không thực sự có trách nhiệm.
- *Phủ cấp 1*: kiểm thử sao cho mỗi lệnh được thực thi ít nhất 1 lần.
- *Phủ cấp 2*: kiểm thử sao cho mỗi điểm quyết định đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE. Ta gọi mức kiểm thử này là phủ các nhánh (Branch coverage). Phủ các nhánh đảm bảo phủ các lệnh.

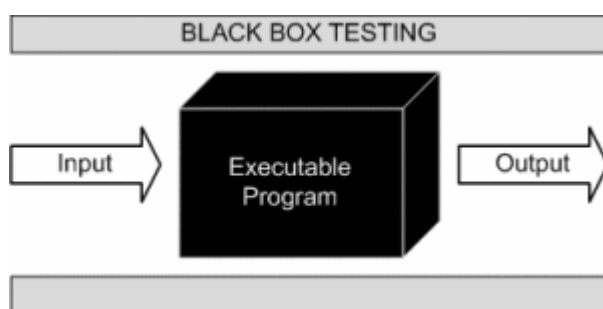
- *Phủ cấp 3*: kiểm thử sao cho mỗi điều kiện luận lý con (subcondition) của từng điểm quyết định đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE. Ta gọi mức kiểm thử này là phủ các điều kiện con (subcondition coverage). Phủ các điều kiện con chưa chắc đảm bảo phủ các nhánh.
- *Phủ cấp 4*: kiểm thử sao cho mỗi điều kiện luận lý con (subcondition) của từng điểm quyết định đều được thực hiện ít nhất 1 lần cho trường hợp TRUE lẫn FALSE & điểm quyết định cũng được kiểm thử cho cả 2 nhánh. Ta gọi mức kiểm thử này là phủ các nhánh & điều kiện con (branch & subcondition coverage).

5.3. Kỹ thuật kiểm thử hộp đen (Black-Box Testing)

Kiểm thử hộp đen: là một phương pháp kiểm thử phần mềm được thực hiện mà không biết được cấu tạo bên trong của phần mềm, là cách mà các tester kiểm tra xem hệ thống như một chiếc hộp đen, không có cách nào nhìn thấy bên trong của cái hộp.

Phương pháp này được đặt tên như vậy bởi vì các chương trình phần mềm, trong con mắt của các tester, giống như một hộp đen; bên trong mà người ta không thể nhìn thấy. Phương pháp này cố gắng tìm ra các lỗi trong các loại sau:

- Chức năng không chính xác hoặc thiếu.
- Lỗi giao diện.
- Lỗi trong cấu trúc dữ liệu hoặc truy cập cơ sở dữ liệu bên ngoài.
- Hành vi hoặc hiệu suất lỗi.
- Khởi tạo và chấm dứt các lỗi.



Hình 1-4: Minh họa Kiểm thử hộp đen

Ưu điểm:

- Kỹ sư kiểm thử có thể không phải IT chuyên nghiệp.
- Hệ thống thật sự với toàn bộ yêu cầu của nó được kiểm thử chính xác.
- Thiết kế kịch bản kiểm thử khá nhanh, ngay khi mà các yêu cầu chức năng được xác định.

Nhược điểm:

- Dữ liệu đầu vào yêu cầu một khối lượng mẫu (sample) khá lớn.
- Khó viết kịch bản kiểm thử do cần xác định tất cả các yếu tố đầu vào, và thiếu cả thời gian cho việc tập hợp này.
- Khả năng để bản thân kỹ sư lạc lối trong khi kiểm thử là khá cao.

Mọi kỹ thuật nào cũng có ưu điểm và nhược điểm của nó. Các hệ thống thường phải được sử dụng nhiều phương pháp kiểm thử khác nhau để đảm bảo được chất lượng của hệ thống khi đến tay người dùng.

6. Kỹ thuật thiết kế Ca kiểm thử

Quá trình phát triển ca kiểm thử có thể giúp tìm ra lỗi trong các yêu cầu hoặc thiết kế của ứng dụng, vì nó đòi hỏi phải tư duy hoàn toàn thông qua các hoạt động của ứng dụng. Vì lý do này, việc chuẩn bị ca kiểm thử sớm nhất có thể trong quy trình phát triển phần mềm là rất hữu ích. Các trường hợp kiểm thử phải bao phủ được toàn bộ luồng xử lý chức năng mô tả trong tài liệu phân tích và thiết kế; các yêu cầu về bảo mật an toàn thông tin, yêu cầu hiệu năng của hệ thống.

6.1. Cấu trúc của Ca kiểm thử

- **Test Case ID:** Giá trị cần để xác định số lượng trường hợp cần để kiểm thử.
- **Testcase Description:** Mô tả sơ lược về mục đích của ca kiểm thử đó.
- **PreRequisites:** Điều kiện tiên đề nếu có.
- **Test Data:** Những dữ liệu đầu vào cần chuẩn bị để test.
- **Step:** Các bước thực hiện 1 ca kiểm thử.
- **Execution Step:** Mô tả các bước thực hiện kiểm thử.
- **Expected results:** Kết quả mong đợi từ các bước thực hiện trên.
- **Actual result:** Kết quả thực tế khi chạy chương trình.
- **Result:** Đánh giá về kết quả, thông thường sẽ là pass, fail.
- **Note:** Cột này dùng để ghi chú những thông tin liên quan khi thực hiện ca kiểm thử.

Các bước xác định Ca kiểm thử:

Bước 1: Xác định mục đích kiểm thử: cần hiểu rõ đặc tả yêu cầu của khách hàng.

Bước 2: Xác định chức năng cần kiểm tra: cần phải biết làm thế nào phần mềm được sử dụng bao gồm các hoạt động, tổ chức chức năng khác nhau.

Các bước thực hiện chỉ mô tả các bước thực hiện đứng từ phía người dùng cuối bao gồm nhập dữ liệu, nhấn button, v.v.

Bước 3: Xác định các yêu cầu phi chức năng: yêu cầu phần cứng, hệ điều hành, các khía cạnh an ninh.

Bước 4: Xác định biểu mẫu cho Ca kiểm thử: bao gồm giao diện UI, chức năng, khả năng tương thích và hiệu suất.

Bước 5: Xác định tính ảnh hưởng giữa các nguyên tắc mô-đun: mỗi một ca kiểm thử nên được thiết kế để có thể che phủ được sự ảnh hưởng của các mô-đun với nhau ở mức độ cao nhất.

Dưới đây là minh họa của một ca kiểm thử (Hình 1.5).

TESTCASE									
Testcase ID	Testcase Description	PreRequisites	Step	Execution Step	Expected Results	Actual Result	Result	Note	
1	Verify Size of the Image in form Ho So.	1. Open form Ho So. 2. Update image in form.	1	Open form Ho So	1. Form Ho So should display on the screen on top. 2. The ImageBox should display on top right of the form. 3. Update button should display under the image.	It is doing the same	Pass		
			2	Click on Update button	1. Form OpenFileDialog should display. 2. Form OpenFileDialog only allows to choose image file with format jpg or png.				
			3	Choose an image	Form OpenFileDialog doesn't allow to choose images which is larger than 200 kilobytes.				
			4	Choose open the image	The chosen image should display in ImageBox on Form Ho So instead the old image.				

Hình 1-5: Minh họa của một ca kiểm thử

6.2. Phân vùng tương đương

6.2.1. Phương pháp

Phân vùng tương đương là phương pháp chia các điều kiện đầu vào thành những vùng tương đương nhau. Tất cả các giá trị trong một vùng tương đương sẽ cho một kết quả đầu ra giống nhau [5]. Vì vậy chúng ta có thể test một giá trị đại diện trong vùng tương đương.

Mục đích: Giảm đáng kể số lượng ca kiểm thử cần phải thiết kế vì với mỗi lớp tương đương ta chỉ cần test trên các phần tử đại diện.

Thiết kế ca kiểm thử bằng kỹ thuật phân vùng tương đương tiến hành theo 2 bước:

1. *Xác định các lớp tương đương:* ta chia miền dữ liệu kiểm thử thành các miền con sao cho dữ liệu trong mỗi miền con có cùng tính chất đối với chương trình. Sau khi chia miền dữ liệu của chương trình thành các miền con

tương đương, ta chỉ cần chọn một phần tử đại diện của mỗi miền con này làm bộ dữ liệu kiểm thử. Các miền con này chính là các lớp tương đương.

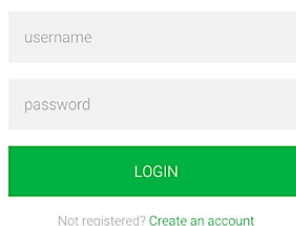
2. Xây dựng các ca kiểm thử tương ứng với mỗi lớp tương đương.

6.2.2. Ví dụ

Ví dụ về 1 Form đăng nhập bao gồm:

username: Text-box

password: Text-box



The image shows a login form with two text input fields. The first field is labeled 'username' and the second is labeled 'password'. Below the fields is a green button with the text 'LOGIN'. At the bottom of the form, there is a link that says 'Not registered? Create an account'.

Hình 1-6: Minh họa một Form đăng nhập

Phân vùng tương đương là phương pháp chia các điều kiện đầu vào thành những vùng tương đương nhau. Tất cả các giá trị trong một vùng tương đương sẽ cho một kết quả đầu ra giống nhau. Vì vậy chúng ta có thể test một giá trị đại diện trong vùng tương đương.

Yêu cầu:

- Thiết kế ca kiểm thử sao cho người dùng nhập vào ô text-box username chỉ cho nhập ký tự chữ với độ dài trong khoảng [6-20].
- Nếu nhập giá trị với số ký tự không nằm trong khoảng [6-20] => hiển thị lỗi “Bạn chỉ được phép nhập chuỗi từ 6 => 20 ký tự”.
- Nếu để trống ô hoặc nhập ký tự khác ký tự chữ => hiển thị lỗi “Tên người dùng chưa hợp lệ! Vui lòng nhập ký tự chữ”.

Dựa vào yêu cầu bài toán ta có thể có các lớp tương đương (phân vùng) sau:

- *Phân vùng 1*: Nhập giá trị hợp lệ từ 6 => 20
- *Phân vùng 2*: Nhập giá trị không hợp lệ < 6 ký tự
- *Phân vùng 3*: Nhập giá trị không hợp lệ > 20 ký tự
- *Phân vùng 4*: Trường hợp để trống không nhập gì hay nhập ký tự không phải dạng chữ

Sau khi áp dụng phân vùng tương đương có thể chọn được các ca kiểm thử sau:

- **Case 1**: Nhập giá trị từ 6 => 20 => pass.
- **Case 2**: Nhập giá trị < 6 ký tự (có thể chọn nhập 1, 2, 3, 4 hoặc 5 ký tự) => hiển thị lỗi “Bạn chỉ được phép nhập chuỗi từ 6 => 20 ký tự”.
- **Case 3**: Nhập giá trị > 20 ký tự (có thể chọn nhập 21, 22, 23,... ký tự) => hiển thị lỗi “Bạn chỉ được phép nhập chuỗi từ 6 => 20 ký tự”.
- **Case 4**: Để trống không nhập gì hay nhập ký tự không phải dạng chữ => hiển thị lỗi “Tên người dùng chưa hợp lệ! Vui lòng nhập ký tự chữ”.

6.2.3. Ưu, nhược điểm

❖ Ưu điểm:

Vì mỗi vùng tương đương ta chỉ cần kiểm tra trên các phần tử đại diện nên số lượng ca kiểm thử được giảm đi khá nhiều nhờ đó mà thời gian thực hiện kiểm thử cũng giảm đáng kể.

❖ Nhược điểm:

Không phải với bất kỳ bài toán nào đều có thể áp dụng kỹ thuật này. Có thể bị thiếu sót lỗi ở biên nếu chỉ chọn giá trị ở khoảng giữa của miền tương đương.

6.3. Phân tích giá trị biên

6.3.1. Phương pháp

Hầu hết các lỗi được tìm thấy khi kiểm tra ở các giá trị biên. Vì vậy phương pháp này tập trung vào việc kiểm thử các giá trị biên này.

Đây là phương pháp test mà chúng ta sẽ test tất cả các giá trị ở vùng biên của dữ liệu vào và dữ liệu ra. Chúng ta sẽ tập trung vào các giá trị biên chứ không test toàn bộ dữ liệu. Thay vì chọn nhiều giá trị trong lớp đương tương để làm đại diện, phân tích giá trị biên yêu cầu chọn một hoặc vài giá trị là các cạnh của lớp tương đương để làm điều kiện test.

Phân tích giá trị biên là trường hợp đặc biệt của phân vùng tương đương, dựa trên những phân vùng tương đương kiểm thử viên sẽ xác định giá trị biên giữa những phân vùng này và lựa chọn ca kiểm thử phù hợp. Mục tiêu là lựa chọn các ca kiểm thử để thực thi giá trị biên.

Các case chuẩn được lựa chọn dựa vào quy tắc sau:

- Giá trị biên nhỏ nhất – 1
- Giá trị biên nhỏ nhất
- Giá trị biên lớn nhất
- Giá trị biên lớn nhất + 1

Nhưng nếu bạn muốn kiểm tra sâu hơn thì bạn cũng có thể lựa chọn theo quy tắc:

- Giá trị biên nhỏ nhất – 1
- Giá trị biên nhỏ nhất
- Giá trị biên nhỏ nhất + 1
- Giá trị biên lớn nhất – 1
- Giá trị biên lớn nhất
- Giá trị biên lớn nhất + 1

6.3.2. Ví dụ

Vấn lý Form đăng nhập và yêu cầu ở phần 7.1.

Theo phương pháp phân vùng tương đương ở trên ta xây dựng được các miền tương đương:

- *Phân vùng 1*: Nhập giá trị hợp lệ từ 6 => 20.
- *Phân vùng 2*: Nhập giá trị không hợp lệ < 6 ký tự.
- *Phân vùng 3*: Nhập giá trị không hợp lệ > 20 ký tự.
- *Phân vùng 4*: Trường hợp để trống không nhập gì hay nhập ký tự không phải dạng chữ.

Áp dụng kỹ thuật phân tích giá trị biên ta chọn được các case sau:

- **Case 1**: Nhập giá trị với 5 ký tự => hiển thị lỗi “Bạn chỉ được phép nhập chuỗi từ 6 => 20 ký tự”.
- **Case 2**: Nhập giá trị với 6 ký tự => pass.
- **Case 3**: Nhập giá trị với 20 ký tự => pass.
- **Case 4**: Nhập giá trị với 21 ký tự => hiển thị lỗi “Bạn chỉ được phép nhập chuỗi từ 6 => 20 ký tự”.
- **Case 5**: Để trống không nhập gì hay nhập ký tự không phải dạng chữ => hiển thị lỗi “Tên người dùng chưa hợp lệ! Vui lòng nhập ký tự chữ.

6.3.3. Ưu, nhược điểm

❖ Ưu điểm:

Thay vì phải kiểm tra hết toàn bộ các giá trị trong từng vùng tương đương, kỹ thuật phân tích giá trị biên tập trung vào việc kiểm thử các giá trị biên của miền giá trị đầu vào để thiết kế ca kiểm thử do “lỗi thường tiềm ẩn tại các ngõ ngách và tập hợp tại biên” nên sẽ tiết kiệm thời gian thiết kế ca kiểm thử và thực hiện kiểm thử.

❖ Nhược điểm:

Phương pháp phân tích giá trị biên chỉ hiệu quả trong trường hợp các đối số đầu vào độc lập với nhau và mỗi đối số đều có một miền giá trị hữu hạn.

6.4. Đoán lỗi

6.4.1. Phương pháp

Trong kiểm thử phần mềm, đoán lỗi là một phương pháp kiểm thử, trong đó các trường hợp kiểm thử được sử dụng để tìm lỗi trong các chương trình đã được phát triển dựa vào kinh nghiệm trong các lần kiểm thử trước. Phạm vi của các trường hợp kiểm thử thường được dựa vào các kiểm thử viên có kiến thức liên quan, là những người đã có kinh nghiệm sử dụng và trực giác để xác định những tình huống thường gây ra lỗi trong phần mềm. Các lỗi điển hình như chia cho không, null pointer, hoặc các biến không hợp lệ, v.v.

Phương pháp đoán lỗi không có quy tắc rõ ràng, ca kiểm thử có thể được thiết kế tùy thuộc vào tình hình, hoặc hoặc luồng công việc trong các tài liệu mô tả chức năng hoặc khi một lỗi không mong muốn / không được mô tả trong tài liệu được tìm thấy trong khi hoạt động kiểm thử. Phương pháp này chỉ phù hợp với những kiểm thử viên có kinh nghiệm. Khi một kiểm thử viên được đưa cho một chương trình, họ phỏng đoán dựa vào trực giác, dựa vào kinh nghiệm, dữ liệu lịch sử về các lỗi đã từng xảy ra với chương trình trước đó, v.v. và sau đó viết các ca kiểm thử để đưa ra các lỗi đó.

6.4.2. Ưu, nhược điểm

❖ Ưu điểm:

Sử dụng phương pháp đoán lỗi có thể giúp kiểm thử viên tìm ra những lỗi điển hình thường xảy ra trong phần mềm hoặc những lỗi không thể tìm thấy khi thiết kế ca kiểm thử theo hình thức thông thường.

❖ Nhược điểm:

Phương pháp đoán lỗi thường được thực hiện bởi các kiểm thử viên có kinh nghiệm và không theo một quy tắc nhất định, thiết kế ca kiểm thử dựa nhiều vào cảm tính.

7. Tạo Bug report

Bug report là một phần rất quan trọng và không thể thiếu trong quy trình thực hiện kiểm thử. Khi phần mềm xảy ra lỗi, kiểm thử viên phải tạo được ra các Bug report và gửi cho nhà phát triển phần mềm đó. Một Bug report được viết rõ ràng và rành mạch, sẽ luôn gây ấn tượng và hiệu ứng tốt hơn đối với một Bug report sơ xài và cầu thả. Làm cho người sửa bug đó và cả người xác nhận lại bug đó không có cảm giác khó chịu khi phải đọc một Bug report sơ xài.

7.1. Bug và Bug report

Bug: Bug của phần mềm là những sai lầm, hỏng hóc, lỗi, khiếm khuyết để tạo ra một kết quả sai, hoặc không lường đến được [6], có thể coi nó như một thứ gì đó không hoạt động đúng theo thiết kế.

Bug report: Văn bản chứa đầy đủ các thông tin về một lỗi của một sản phẩm được kiểm thử viên gửi cho một tổ chức hay cá nhân liên quan để sửa được gọi là Bug report.

7.2. Cấu trúc một Bug report

- **Project**: tên của dự án phần mềm.
- **Reported by**: kiểm thử viên tạo ra Bug report.
- **Bug Name, Bug ID và Date**: tên của bug, ID và ngày tạo report.
- **Assigned to**: cá nhân hoặc tổ chức phát triển phần mềm đó.
- **Status**: Trạng thái thực hiện của report.
- **Summary/Description**: mô tả ngắn gọn về bug.
- **Environments (OS/Browser)**: môi trường chạy thử phần mềm.
- **Step to reproduce**: mô tả lại các bước thực hiện gây ra bug.

- **Actual results:** kết quả thực tế.
- **Expected results:** kết quả mong đợi.
- **Severity:** mức độ nghiêm trọng của bug.
- **Priority:** mức độ ưu tiên của bug.
- **Attachment:** đính kèm với bug (tệp, đường dẫn URL, ảnh, v.v.)

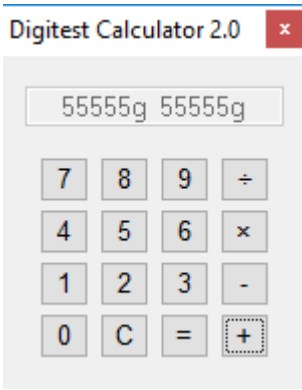
Một Bug Report được minh họa đầy đủ trong Hình 1.7.

BUG REPORTS

Project: Calculator

Reported by: Van Hoang, Pham

- **Bug Name:** Plus button clickdown
Bug ID: Cal0001
- **Date:**25-Oct-16
- **Assigned to:** Developer-TEAM1016
- **Status:** New, retest
- **Summary/Description:**
This bug insert a string “55555g” to the TextBox instead of correct operator “+”.
- **Environments (OS/Browser):** Windows 10 /64bit/Core I7/ Ram 8GB/...
- **Step to reproduce:**
 1. Click on the “Plus” sign button, the program allow user click on the Plus button, but
 2. The TextBox displays incorrect sign. The TextBox displays string “55555g” many times.
- **Actual results:** The TextBox display the result “55555g” after click on Plus sign button.
- **Expected results:** The TextBox displays only plus sign before a number or between two number in an expression
- **Severity:** Critical (S1)
Priority: High (P1)
- **Attachment:**



Hình 1-7: Minh họa một Bug report

Một số yêu cầu khi tạo Bug report:

- Tiêu đề phải rõ ràng: Khi lập trình viên đọc bug, thứ đầu tiên đập vào mắt là Bug name. Nó cũng là phần được đọc nhiều nhất, không phải là description. Một bug name tốt phải ngắn gọn và diễn tả được bug một cách tối giản.
- Phải mô phỏng lại được quá trình gây ra bug: nếu không mô phỏng lại được, bug sẽ không thể được khắc phục.
- Không viết luận trong description: viết ngắn gọn và vào trọng tâm. Cố gắng viết ít chữ nhất có thể nhưng vẫn đầy đủ ý.

7.3. Severity và Priority

Có hai phân quan trọng trong những bug report đó là [7]:

- *Severity* - Mức độ nghiêm trọng
- *Priority* - Mức độ ưu tiên

Mặc dù hai yếu tố này không phải là yếu tố sống còn trong quản lý bug. Tuy nhiên, việc hiểu đúng về mức độ nghiêm trọng, độ ưu tiên của sản phẩm cho thấy chúng ta thực sự hiểu rõ và quan tâm đến chất lượng sản phẩm cũng như thể hiện sự chuyên nghiệp của một kỹ sư kiểm thử.

7.3.1. Severity

Severity là mức độ mà các bug có thể ảnh hưởng đến các phần mềm. Nói cách khác, nó xác định các tác động mà một bug nhất định có trên hệ thống.

Severity có thể được phân thành các loại sau đây [8]:

- **Critical (S1):** Bug ảnh hưởng đến các chức năng quan trọng hoặc dữ liệu quan trọng. Nó không có giải pháp để thay thế. Ví dụ: Cài đặt không thành công, hoàn thành sự thất bại của một tính năng, v.v.
- **Major (S2):** Thiệt hại ảnh hưởng đến chức năng chính hoặc dữ liệu chính. Nó có giải pháp để thay thế nhưng không rõ ràng hoặc khó khăn.

Ví dụ: Một tính năng không thể thực thi trực tiếp nhưng là khả thi nếu có 10 bước gián tiếp phức tạp được thực hiện để có được kết quả như mong muốn.

- **Minor (S3):** Bug ảnh hưởng đến chức năng nhỏ hoặc dữ liệu không quan trọng. Nó có một giải pháp thay thế dễ dàng. Ví dụ: Một tính năng nhỏ không được thực thi nhưng nhiệm vụ tương tự có thể dễ dàng thực hiện từ một chức năng khác.
- **Trivial (S4):** Bug không ảnh hưởng đến chức năng hoặc dữ liệu. Nó thậm chí không cần một giải pháp để thay thế do không ảnh hưởng đến năng suất hoặc hiệu quả mà chỉ là sự bất tiện. Ví dụ: Sai lệch bố cục nhỏ, lỗi chính tả / lỗi ngữ pháp, v.v.

7.3.2. Priority

Priority xác định thứ tự mà chúng ta nên giải quyết một bug. Chúng ta nên sửa nó ngay bây giờ, hoặc nó có thể được hoãn lại cho đến khi một bug nghiêm trọng khác đã được giải quyết. Tình trạng ưu tiên này được thiết lập bởi các kiểm thử viên cho nhà phát triển để cập đến các khung thời gian để sửa chữa những bug. Mức độ ưu tiên càng cao thì phải sửa chữa nó trong thời gian càng sớm. Tình trạng ưu tiên được thiết lập dựa trên các yêu cầu của khách hàng.

Priority có thể phân thành các loại sau đây:

- **Urgent (P0):** Phải được sửa càng sớm càng tốt.
- **High (P1):** Phải được sửa trong một vài phiên bản tiếp theo.
- **Medium (P2):** Nên được sửa ở những phiên bản tiếp theo.
- **Low (P3):** Có thể được sửa ở một phiên bản nào đó.

CHƯƠNG 2: KIỂM THỬ TRÊN THIẾT BỊ DI ĐỘNG

Chương này sẽ giới thiệu sơ lược về lĩnh vực kiểm thử trên di động, đặc biệt là về kiểm thử tự động trên di động. Tuy nhiên, về cơ bản thì kiểm thử tự động trên mọi nền tảng đều là để tự động hóa quá trình kiểm thử, vì vậy phạm vi khóa luận sẽ chỉ tập trung vào giới thiệu về kiểm thử tự động nói chung. Ngoài ra, sự khác nhau giữa kiểm thử trên di động so với trên các nền tảng khác và ưu, nhược điểm của kiểm thử tự động cũng sẽ được đề cập đến.

1. Kiểm thử trên thiết bị di động

Như chúng ta đã biết thì Công nghệ điện thoại di động và các thiết bị thông minh hiện nay là xu hướng và cũng là tương lai của thế giới. Mỗi ngày có hàng triệu ứng dụng được tải xuống từ Appstore hoặc Google Play về các thiết bị cá nhân. Các ứng dụng di động rất phong phú đa dạng đáp ứng đủ các nhu cầu học tập, chăm sóc sức khỏe hay giải trí của người dùng. Để kiểm thử được các ứng dụng trên các thiết bị di động (Mobile App Testing) thì trước hết ta cần hiểu định nghĩa về các thiết bị di động.

1.1. Các khái niệm cơ bản về ứng dụng di động

1.1.1. Giới thiệu

Ngày nay điện thoại thông minh và máy tính bảng là những thiết bị không thể thiếu đối với mỗi chúng ta. Những thiết bị này hiện nay đã có cả tỷ người sử dụng. Vì chúng rất phổ biến nên chúng phải có sự tin cậy, bảo mật và có tính tương thích cao. Khi việc dùng của chúng tăng lên thì nhu cầu về nhiều ứng dụng cũng tăng lên. Đó là lí do tại sao phát triển và kiểm thử ứng dụng di động là việc làm không thể thiếu trong công nghiệp công nghệ thông tin.

Điện thoại thông minh về căn bản là tổ hợp của máy tính và điện thoại cho nên nó là thiết bị phức tạp hơn máy tính. Có khác biệt giữa kiểm thử ứng

dụng di động và kiểm thử ứng dụng máy tính. Nhiều người nghĩ phần mềm là phần mềm, nếu tôi có thể kiểm thử phần mềm trên máy tính, tôi có thể kiểm thử phần mềm trên điện thoại thông minh. Mặc dù các nguyên lý kiểm thử là như nhau nhưng kỹ thuật là khác nhau và yêu cầu cũng nhiều hơn. Trước khi xây dựng hay kiểm thử ứng dụng di động, bạn cần biết rằng sẽ có hàng triệu người dùng nó. Nếu đưa ra một ứng dụng di động với nhiều lỗi, nó có thể là một thảm họa.

Điện thoại di động có bộ nhớ giới hạn và năng lực xử lý giới hạn, cho nên điều quan trọng là kiểm thử cách thiết bị làm việc khi nó đang đầy năng lực. Điều cũng quan trọng là nghĩ về tuổi thọ của pin liệu pin có hết chóng hơn với ứng dụng của bạn chạy không và điều gì xảy ra khi pin hết? Bạn phải kiểm thử cả tính dùng được. Tính dùng được nghĩa là kiểm thử nó trên người dùng thực về cách họ tương tác với ứng dụng? Dùng ứng dụng dễ thế nào? Các ứng dụng khó dùng thường bị xoá đi một khi người dùng thấy khó dùng. Bạn cần biết cách ứng dụng khớp với màn hình nhỏ. Chữ có dễ đọc không? Cách ứng dụng của bạn trông trên màn hình nhỏ là rất quan trọng. Ứng dụng có chạy nhanh không? Người dùng có cảm thấy họ đang đợi quá lâu để cho một yêu cầu được đáp ứng?

1.1.2. Phân loại ứng dụng trên thiết bị di động

❖ Ứng dụng gốc (Native applications)

Ứng dụng được thiết kế đặc biệt chỉ chạy trên một hệ điều hành của một thiết bị nào đó và thường phải điều chỉnh để chạy được trên các thiết bị khác nhau. Native App, được hiểu nôm na là ứng dụng gốc, hay ứng dụng được viết cho các thiết bị di động, chạy trên từng nền tảng (iOS, Android, WindowsPhone, v.v.) khác nhau và tất nhiên là trên các thiết bị khác nhau để thực hiện một chức năng cụ thể như: danh bạ, lịch, phần mềm nghe nhạc, xem video trên điện thoại/tablet, v.v. và đa số các trò chơi trên thiết bị di động đều là ứng dụng gốc.

❖ *Ứng dụng Web (Web applications)*

Ứng dụng được xây dựng trên nền tảng web chuẩn, được phát triển sử dụng công nghệ truyền thống web (HTML, CSS, Javascript) và viết mã lệnh phía máy chủ trong Node.js, PHP, ASP.NET, v.v. Các ứng dụng web phù hợp cho trình duyệt trên điện thoại thông minh hoặc máy tính bảng và người dùng cần một trình duyệt và kết nối Internet để sử dụng chúng.

❖ *Ứng dụng lai (Hybrid applications)*

Là sự kết hợp giữa ứng dụng gốc và ứng dụng Web, thường sử dụng kỹ thuật làm web như HTML5, CSS. Đây là một sự kết hợp của các công nghệ trước. Ứng dụng lai thường được xây dựng đa nền tảng vì vậy đặc tính chính và lợi thế của chúng là tính di động. Có một số phương pháp bàn đến vấn đề làm thế nào để xây dựng các ứng dụng lai. Hầu hết sử dụng đa nền tảng ứng dụng công nghệ web. Sự khác biệt giữa ứng dụng lai và ứng dụng web là ứng dụng lai có gói ứng dụng gốc và nó thường được cài đặt từ nền tảng AppStore riêng.

1.1.3. Các hệ điều hành trên thiết bị di động

❖ *Hệ điều hành Android*

Android là hệ điều hành phát triển nhanh nhất và phổ biến rộng rãi nhất trong các hệ điều hành di động. Android được sở hữu và quản lý bởi Open Handset Alliance - tập đoàn công nghiệp tạo phần cứng, phần mềm và viễn thông tiêu chuẩn mở cho thiết bị di động. Tập đoàn này được dẫn dắt bởi Google. Kể từ khi Android là dự án mã nguồn mở dựa trên Linux, hầu hết các nhà sản xuất và các hãng di động tận dụng điều đó và sửa đổi hệ điều hành cho phù hợp với phần cứng của họ, tăng sự phức tạp của các hệ thống. Thực tế này làm cho hệ điều hành điện thoại di động Android bị phân mảnh nhất, làm tăng chi phí kiểm tra và phức tạp. Tuy nhiên, nhiều công cụ kiểm thử khác nhau và mục tiêu các framework mục chủ yếu là cho Android do nó là hệ điều hành phổ biến nhất.

❖ *Hệ điều hành iOS*

iOS là hệ điều hành di động phát triển và sở hữu bởi Apple Inc. Đó là mã nguồn đóng, hệ thống vận hành giống Unix dựa trên Darwin (BSD) và OS X. iOS được phát triển cho iPhone, nhưng bây giờ iOS chạy trên iPad, iPod Touch hoặc Apple TV. Các nhà sản xuất khác không được cấp phép sử dụng iOS. Do đó, chỉ các thiết bị của Apple có thể chạy nó.

Hầu hết người dùng iOS có phiên bản iOS 10 hay mới nhất 11. Kết quả là, thử nghiệm trên tất cả các thiết bị có sẵn không phải là khó khăn như trên Android. Hệ điều hành phân mảnh được trình bày chủ yếu bởi các tính năng thiết lập hệ điều hành khác nhau (một số tính năng không có sẵn trên một số thiết bị). Mặt khác, các thiết bị của Apple là một trong những thiết bị rất tốn kém.

Tiêu chuẩn tích hợp môi trường phát triển (IDE) cho các ứng dụng gốc là Xcode và chỉ có thể được cài đặt trên hệ điều hành của Apple OS X.

❖ *Hệ điều hành Windows & Windows Phone*

Với Windows 8, Microsoft đã chuyển hệ điều hành Windows đến với các thiết bị di động. Windows 8.1, là phiên bản hiện tại, có thể chạy trên máy tính cá nhân và máy tính bảng. Hơn nữa, Microsoft đã có hệ điều hành Windows Phone đặc biệt cho điện thoại thông minh. Hai nền tảng được hội tụ trong Windows 10.

1.2. Phương pháp kiểm thử trên thiết bị di động

1.2.1. Kiểm thử trên các thiết bị thực

Kiểm thử trên các thiết bị thực là thực sự cần thiết cho tất cả các ứng dụng di động. Nó cho các kết quả thực tế nhất và kiểm thử có thể thực hiện trên tất cả các kịch bản kiểm thử cần thiết. Mặt khác, nó vô cùng khó khăn và tốn kém để kiểm thử ứng dụng trên tất cả các thiết bị có sẵn.

❖ Ưu điểm:

Giao diện tương tác người dùng thật, để đạt được thử nghiệm người dùng đáng tin cậy, xử lý một thiết bị thực là thực sự cần thiết.

Tương tác với cảm biến - nếu ứng dụng thử nghiệm sử dụng cảm biến (ví dụ gia tốc, la bàn, máy ảnh, v.v.), sau đó thử nghiệm đã vận hành thiết bị trong tay. Một số mô phỏng cũng bắt chước được hành vi của các cảm biến này nhưng kết quả của nó không thực tế.

Độ tin cậy - tất cả các trường hợp kiểm thử trên các thiết bị thực tế cho chúng ta sự bảo đảm tốt nhất giữa người sử dụng với các thiết bị tương ứng.

❖ Nhược điểm

Loại thiết bị và chi phí thử nghiệm trên vài thiết bị thực không bảo đảm ứng dụng sẽ làm việc trên thiết bị khác. Mua thiết bị kiểm thử mới cũng rất tốn kém.

Giới hạn nhà cung cấp dịch vụ di động - khó thử nghiệm các ứng dụng di động mà phải thực hiện trên các nhà cung cấp dịch vụ di động khác nhau vì khác cơ sở hạ tầng mạng hoặc có thể bị cấm bởi một số trang web.

1.2.2. Kiểm thử trên máy mô phỏng và giả lập

Máy mô phỏng và giả lập là loại phần mềm cho phép chạy một hệ thống máy tính trên nền tảng máy chủ.

Hiện tại có sẵn cho mỗi nền tảng và thường tích hợp vào IDE như Android Studio, Xcode hoặc Visual Studio.

Có sự khác biệt giữa giả lập và mô phỏng. Giả lập bắt chước hành vi của phần mềm và sử dụng tất cả tài nguyên sẵn có từ máy chủ như CPU, bộ nhớ, mạng, v.v. Kiểm thử trên giả lập có thể cung cấp kết quả sai lệch bởi vì máy chủ có thể nhanh hơn nhiều so với các thiết bị thật.

Chương trình mô phỏng cung cấp kết quả thực tế hơn vì chúng thể hiện thiết bị một cách chính xác hơn. Tốc độ xử lý, sử dụng bộ nhớ hoặc sự hiện diện thẻ nhớ, thậm chí các loại kết nối mạng hoặc cường độ tín hiệu có thể được thiết lập để tạo ra môi trường thực tế hơn.

Có rất nhiều mô phỏng và giả lập trên thị trường với các tính năng khác nhau và hạn chế. Tuy nhiên, tất cả đều có một số ưu điểm và nhược điểm tương tự.

❖ *Ưu điểm:*

Chi phí thấp - giả lập chuẩn có trong bộ cài cùng với SDK là miễn phí.

Luôn được cập nhật phiên bản mới của giả lập cùng với phiên bản mới của SDK.

Nhanh chóng và đơn giản để sử dụng - ứng dụng có thể dễ dàng triển khai và cài đặt từ IDE.

Tùy chọn vị trí, kích thước màn hình và độ phân giải, loại kết nối, máy ảnh, v.v. có thể được mô phỏng.

❖ *Nhược điểm:*

Kết quả thử nghiệm bị sai lệch do giả lập và luôn luôn có một khả năng mà các ứng dụng có thể xử lý hơi khác nhau trên thiết bị thực tế bởi vì các phần cứng khác nhau, môi trường mạng hay phần mềm khác biệt.

Không thể thực hiện trong tất cả trường hợp kiểm thử - khi làm việc với giả lập, thử nghiệm sẽ bị giới hạn bởi tài nguyên máy chủ và tính năng bộ mô phỏng. Ứng dụng thử nghiệm sử dụng kết nối không dây hoặc la bàn có thể không được thực hiện cũng như gián đoạn của ứng dụng với cuộc gọi đến hay SMS.

Không đáp ứng kịp thời - một số giả lập chậm và hình ảnh động hay trò chơi bị chậm. Trong trường hợp kiểm thử tài liệu dài, thời gian có thể kéo dài hơn.

1.2.3. Kiểm thử trên thiết bị di động đám mây

Thử nghiệm thiết bị di động đám mây là một dịch vụ cho phép để chạy tự động hoá hoặc thử nghiệm bằng tay trên hàng trăm thiết bị vật lý trong đám mây. Ví dụ như SOASTA CloudTest, SeeTest Cloud, v.v.

Mục đích chính của những thử nghiệm thiết bị di động đám mây là để tìm lỗi mà chỉ xuất hiện trên một số thiết bị hoặc một số phiên bản hệ điều hành. Nó có thể tiết lộ các lỗi có thể được ẩn từ các nhà phát triển nếu họ sử dụng giả lập và chỉ có vài thiết bị vật lý. Ví dụ, giao diện người dùng quá tải (sai lầm phổ biến) hay như một sự chậm trễ ngắn trên thử nghiệm thiết bị vật lý và mô phỏng có thể sẽ nhận thấy. Nhưng trên các thiết bị cũ hoặc cấp thấp với sức mạnh tính toán thấp, nó có thể gây ra hiện tượng các giao diện người dùng không được hiển thị đúng trong trường hợp tốt, treo hoặc sập trong trường hợp tồi tệ hơn.

❖ Ưu điểm:

Đa dạng về thiết bị - Kiểm thử viên phần mềm có thể nhanh chóng phát hiện lỗi hay thiếu sót đó bắt nguồn từ việc sử dụng phần cứng hoặc phần mềm khác nhau trong các thiết bị di động. Không cần phải nâng cấp và hạ cấp thiết bị vật lý của nhà phát triển để thi hành thử nghiệm trên vài phiên bản hệ điều hành.

Khả năng tự động hoá - tự động chạy thử cùng một lúc nhiều thiết bị. Kiểm thử viên phần mềm có thể xem bản ghi thử nghiệm đối với mỗi thiết bị thông qua web tương tác.

❖ Nhược điểm:

Chi phí - điều này có thể là một cản trở cho các công ty nhỏ hoặc cỡ trung mà giải pháp di động không phải là ưu tiên.

Không thể để kiểm tra hết tất cả các trường hợp thử nghiệm - mặc dù các thử nghiệm được thực hiện trên các thiết bị thực tế, ta vẫn không thể kiểm tra được la bàn, gia tốc, GPS hoặc máy ảnh.

Bảo mật - đám mây được dùng chung vậy nên sẽ có một số vấn đề bảo mật xảy ra với việc gửi gói ứng dụng trên đám mây. Ứng dụng sẽ được chạy trong môi trường không an toàn nơi giả lập chạy thử ứng dụng.

Không đáp ứng kịp thời - thời gian thử nghiệm có thể kéo dài vì độ trễ mạng.

1.3. Các loại kiểm thử di động

1.3.1. Kiểm thử phần cứng

Thiết bị bao gồm bộ xử lý trong, phần cứng bên trong, kích thước màn hình, độ phân giải, bộ nhớ, camera, radio, bluetooth, wifi, v.v.

1.3.2. Kiểm thử phần mềm hoặc kiểm thử ứng dụng.

Ứng dụng làm việc trên các thiết bị di động và các chức năng của nó được kiểm thử. Điều này gọi là kiểm thử ứng dụng di động để phân biệt với các nền tảng khác. Ngay cả trong ứng dụng di động, có một vài điểm khác biệt cơ bản quan trọng cần phải hiểu về các loại ứng dụng:

- Ứng dụng gốc chỉ chạy trên một nền tảng trong khi ứng dụng di động web thì đa nền tảng.
- Ứng dụng gốc được viết trên nền tảng giống SDK trong khi ứng dụng di động web được viết với các công nghệ web: html, css, asp.net, java, php.
- Trong một vài ứng dụng gốc, việc cài đặt là bắt buộc nhưng trong ứng dụng di động web, việc cài đặt là không cần thiết.
- Ứng dụng gốc phải được cập nhật từ play store hoặc app store trong khi các ứng dụng di động web thì không cần như vậy.
- Rất nhiều ứng dụng gốc không yêu cầu kết nối internet nhưng ứng dụng di động web thì yêu cầu.

- Ứng dụng gốc làm việc nhanh hơn khi so sánh với ứng dụng di động web
- Ứng dụng gốc được cài đặt thông qua app stores hoặc google play store trong khi ứng dụng mobile web là các websites và được truy nhập thông qua internet.

1.4. Các đặc điểm của kiểm thử di động

1.4.1. Sự đa dạng các thiết bị di động

- Đa dạng các hãng sản xuất thiết bị như HTC, SamSung, Apple, Nokia, v.v. với các kích thước màn hình và cấu hình phần cứng khác nhau.
- Đa nền tảng: Việc chắc chắn rằng ứng dụng di động chạy được trên tất cả các loại thiết bị (smartphone, tablet hay phablet được cung cấp bởi một số các nhà cung cấp lớn (như Samsung, Sony, Nokia, HTC, Apple...) và trên tất cả các hệ điều hành (iOS, Android, Windows, Blackberry...) thực sự là một thách thức.
- Các thiết bị di động có thời gian chạy ứng dụng khác nhau.

1.4.2. Thách thức phần cứng của thiết bị

- Khả năng thích nghi và không gian giới hạn khiến kích thước màn hình thay đổi liên tục: Với mỗi một dòng máy, mỗi loại thiết bị sẽ có nhiều kích thước khác nhau để đáp ứng tính cạnh tranh trên thị trường nên việc phát triển làm sao để ứng dụng có thể thích nghi với kích thước của các màn hình khác nhau cũng là một thách thức.
- Giới hạn tốc độ xử lý: các thiết bị di động đang ngày càng được nâng cấp về tốc độ xử lý để phù hợp với nhu cầu ngày càng cao của người dùng.
- Giới hạn dung lượng bộ nhớ của thiết bị: không giống như trên máy tính, dung lượng bộ nhớ của các thiết bị di động gần như không thể thay thế hay nâng cấp.
- Sự khác biệt về giao thức của thiết bị WAP/HTTP.

1.4.3. Thách thức về đường truyền mạng

- Đa dạng các loại mạng (GSM/GPRS/WIFI/3G).
- Không dự đoán được thời gian cho truyền tải dữ liệu.
- Khác biệt về tốc độ kết nối.
- Đa dạng các nhà mạng với những tính năng mạng khác nhau.

1.4.4. Các ca kiểm thử đặc biệt cho kiểm thử các ứng dụng di động

- *Sự hao tổn pin*: Việc theo dõi sự hao tổn pin khi chạy ứng dụng trên thiết bị di động rất quan trọng.
- *Tốc độ chạy ứng dụng*: Theo dõi thời gian phản hồi trên các thiết bị khác nhau với các dung lượng bộ nhớ khác nhau, tốc độ mạng khác nhau.
- *Yêu cầu bộ nhớ*: Khi tải, cài đặt và chạy ứng dụng
- *Kiểm tra tính tương thích của ứng dụng*: Để đảm bảo ứng dụng không bị crash khi mất kết nối mạng hoặc các tác động ngoại vi khác.

Ngoài ra kiểm thử ứng dụng di động cũng bao gồm các dạng kiểm thử sau:

- *Kiểm thử giao diện (UI Testing)*: Kiểm tra màu sắc, phong cách Menu, nhất quán của giao diện người dùng trên các thiết bị khác nhau.
- *Kiểm thử chức năng (Function Testing)*: Kiểm tra các chức năng chính của ứng dụng di động theo đặc điểm kỹ thuật của thiết bị.
- *Kiểm thử hiệu suất và chịu tải (Performance and Load Test)*: Kiểm tra hành vi của ứng dụng di động trong các nguồn tài nguyên thấp (Bộ nhớ/ Không gian lưu trữ), hành vi của trang web điện thoại di động khi nhiều người sử dụng điện thoại di động cùng truy cập vào trang app di động.
- *Kiểm tra khả năng sử dụng (Usability Testing)*: Kiểm tra các khía cạnh khả năng sử dụng các ứng dụng di động.
- *Thử nghiệm tương thích (Compatibility Testing)*: Kiểm tra khả năng tương thích của ứng dụng của bạn với các tính năng thiết bị gốc để đảm bảo rằng ứng dụng của bạn không cản trở các ứng dụng khác trong thiết bị.

- *Kiểm tra gián đoạn*: Vì lí do các thiết bị di động có bộ nhớ thấp hơn nhiều so với desktop nên phải đảm bảo rằng khi có cuộc gọi thoại, tin nhắn SMS, cảm sạc, thông báo bộ nhớ thấp trong khi ứng dụng đang chạy không gây ra bất cứ xung đột nào.

2. Kiểm thử tự động

2.1. Khái niệm kiểm thử tự động

Kiểm thử tự động là quá trình thực hiện một cách tự động các bước trong một ca kiểm thử. Nó sử dụng một công cụ kiểm thử tự động nào đó để rút ngắn thời gian kiểm thử. Kiểm thử tự động hỗ trợ các kiểm thử viên rất nhiều tùy vào công cụ và các nội dung kiểm thử có thể thực hiện bằng tay hay không. Đối với những nhiệm vụ kiểm tra khó mà thực hiện bằng tay hoặc yêu cầu chi phí về nhân công là quá lớn thì sử dụng công cụ hỗ trợ là điều hết sức cần thiết.

2.2. Mục tiêu của kiểm thử tự động

Phần mềm có khiếm khuyết là thông thường và gây ra thiệt hại về kinh tế theo thời gian. Chính vì vậy các tổ chức về phần mềm dành nhiều thời gian và nguồn lực để phân tích và kiểm thử phần mềm.

Ngày nay ứng dụng tự động hóa vào các ngành đa dạng, trong đó có ngành kiểm thử. Trước đây kiểm thử viên kiểm thử bằng tay thực hiện và ghi lại kết quả trên giấy, nhưng với ứng dụng công nghệ thông tin thì các công cụ kiểm thử cũng phát triển từ rất sớm hỗ trợ kiểm thử viên rất nhiều và đặc biệt là các trường hợp kiểm thử đặc biệt mà kiểm thử bằng tay không thể thực hiện được hoặc rất khó khăn để thực hiện. Các tổ chức càng quan tâm đến chất lượng phần mềm thì càng có nhiều chức năng và nội dung được kiểm tra đòi hỏi kiểm thử viên phải thực hiện nhiều công việc hơn vì vậy kiểm thử với sự hỗ trợ của công cụ là rất cần thiết.

Để giúp các kiểm thử viên có thể kiểm thử tự động đó là các Test Tool, tuy nhiên không phải trong bất cứ trường hợp nào cũng có thể kiểm thử tự động. Vậy kiểm thử tự động khi nào?

- Kiểm thử tự động trong các tình huống sau:
 - Không đủ tài nguyên:
 - Khi số lượng tình huống kiểm tra quá nhiều mà các kiểm thử viên không thể hoàn tất bằng tay trong thời gian cụ thể nào đó.

Có thể lấy một dẫn chứng là khi thực hiện kiểm tra chức năng của một website. Website này sẽ được kiểm tra với 6 môi trường gồm 4 trình duyệt và 2 hệ điều hành. Tình huống này đòi hỏi số lần kiểm tra tăng lên và lặp lại 6 lần so với việc kiểm tra cho một môi trường cụ thể.

- Kiểm tra hồi quy:

Trong quá trình phát triển phần mềm, nhóm lập trình thường đưa ra nhiều phiên bản phần mềm liên tiếp để kiểm tra. Thực tế cho thấy việc đưa ra các phiên bản phần mềm có thể là hàng ngày, mỗi phiên bản bao gồm những tính năng mới, hoặc tính năng cũ được sửa lỗi hay nâng cấp. Việc bổ sung hoặc sửa lỗi mã nguồn cho những tính năng ở phiên bản mới có thể làm cho những tính năng khác đã kiểm tra tốt chạy sai mặc dù phần mã nguồn của nó không hề chỉnh sửa. Để khắc phục điều này, đối với từng phiên bản, kiểm thử viên không chỉ kiểm tra chức năng mới hoặc được sửa, mà phải kiểm tra lại tất cả những tính năng đã kiểm tra tốt trước đó. Điều này khó khả thi về mặt thời gian nếu kiểm tra thủ công.

- Kiểm tra vận hành phần mềm trong môi trường đặc biệt:

Đây là kiểm tra nhằm đánh giá xem vận hành của phần mềm có thỏa mãn yêu cầu đặt ra hay không. Thông qua đó kiểm thử viên có thể xác định được các yếu tố về phần cứng, phần mềm ảnh hưởng đến khả năng vận hành của phần mềm.

➤ *Mục tiêu của kiểm thử tự động:*

- Giảm bớt công sức và thời gian thực hiện.
- Tăng độ tin cậy.
- Giảm sự nhàm chán.
- Giảm chi phí tổng cho quá trình kiểm thử.

➤ *Ưu điểm của kiểm thử tự động:*

- Kiểm thử phần mềm không cần can thiệp của kiểm thử viên.
- Giảm chi phí khi thực hiện kiểm tra số lượng lớn ca kiểm thử hoặc ca kiểm thử lặp lại nhiều lần.
- Giả lập được các tình huống khó có thể thực hiện bằng tay.

2.3. Nguyên tắc kiểm thử tự động

Thực sự là sai lầm khi nghĩ tự động là đơn giản chụp lại, ghi lại một tiến trình kiểm thử thủ công. Thực tế, kiểm thử tự động có những điểm khác với kiểm thử thủ công. Nó có những lỗi và khả năng dự đoán.

Vì thế, những cơ hội thành công với kiểm kiểm thử tự động sẽ được cải thiện đáng kể trong trường hợp bạn thực sự hiểu nó.

Kiểm thử tự động tuân theo đầy đủ những nguyên tắc kiểm thử nói chung, đó là các nguyên tắc sau [9]:

❖ *Nguyên tắc 1 – Kiểm thử đưa ra lỗi:*

Kiểm thử có thể cho thấy rằng phần mềm đang có lỗi, nhưng không thể chứng minh rằng phần mềm không có lỗi. Kiểm thử làm giảm xác suất lỗi chưa tìm thấy vẫn còn trong phần mềm, thậm chí là không còn lỗi nào, nó vẫn không phải là bằng chứng của sự chính xác.

❖ *Nguyên tắc 2 – Kiểm thử mọi thứ là không thể*

Kiểm thử mọi thứ (tất cả các tổ hợp của điều kiện input đầu vào) là không thể thực hiện được, trừ phi nó chỉ bao gồm một số trường hợp bình thường (ít

trường hợp tổ hợp thì có thể test toàn bộ được). Thay vì kiểm thử toàn bộ, việc phân tích rủi ro và dựa trên sự mức độ ưu tiên chúng ta có thể tập trung việc kiểm thử vào một số điểm cần thiết.

❖ *Nguyên tắc 3 – Kiểm thử sớm*

Để tìm được bug sớm, các hoạt động kiểm thử nên được bắt đầu càng sớm càng tốt trong quy trình phát triển (vòng đời phát triển) phần mềm hoặc hệ thống, và nên tập trung vào các hoạt động đã định trước.

❖ *Nguyên tắc 4 – Sự tập trung của lỗi*

Nỗ lực kiểm thử nên tập trung một cách cân đối vào mật độ lỗi dự kiến và lỗi phát hiện ra sau đó trong các mô-đun. Một số ít các mô-đun thường chứa nhiều lỗi không phát hiện ra trong lúc kiểm thử trước khi phát hành, hoặc chịu trách nhiệm cho hầu hết các lỗi hoạt động của phần mềm.

❖ *Nguyên tắc 5 – Nghịch lý thuốc trừ sâu*

Nếu việc kiểm thử tương tự nhau được lặp đi lặp lại nhiều lần, thì cuối cùng sẽ có một số trường hợp kiểm thử sẽ không còn tìm thấy bất kỳ lỗi nào mới. Để khắc phục "nghịch lý thuốc trừ sâu" này, các trường hợp kiểm thử cần phải được xem xét và sửa đổi thường xuyên, và cần phải viết các ca kiểm thử mới và khác nhau để thực hiện nhiều phần khác nhau của phần mềm hoặc hệ thống để tìm ra lỗi tiềm ẩn nhiều hơn nữa.

Nguyên tắc này giống như việc trừ sâu trong nông nghiệp, nếu chúng ta cứ phun một loại thuốc với nồng độ giống nhau trong một khoảng thời gian dài thì có một số con sâu sẽ quen dần và cuối cùng việc phun thuốc giống như là tắm chúng vậy (bị lờn thuốc) => lúc đó chúng ta không thể diệt sạch chúng được. Do vậy, để diệt sạch sâu một cách hiệu quả, người ta thường thay đổi loại thuốc trừ sâu, mỗi loại chỉ dùng trong khoảng thời gian ngắn.

❖ *Nguyên tắc 6 – Kiểm thử theo các ngữ cảnh độc lập*

Nguyên tắc này là việc testing phụ thuộc vào ngữ cảnh, test trong nhiều ngữ cảnh khác nhau.

Để hiểu rõ hơn chúng ta xem ví dụ sau:

Ví dụ: Cùng với một chương trình *calculator* nhưng có rất nhiều phạm vi ứng dụng:

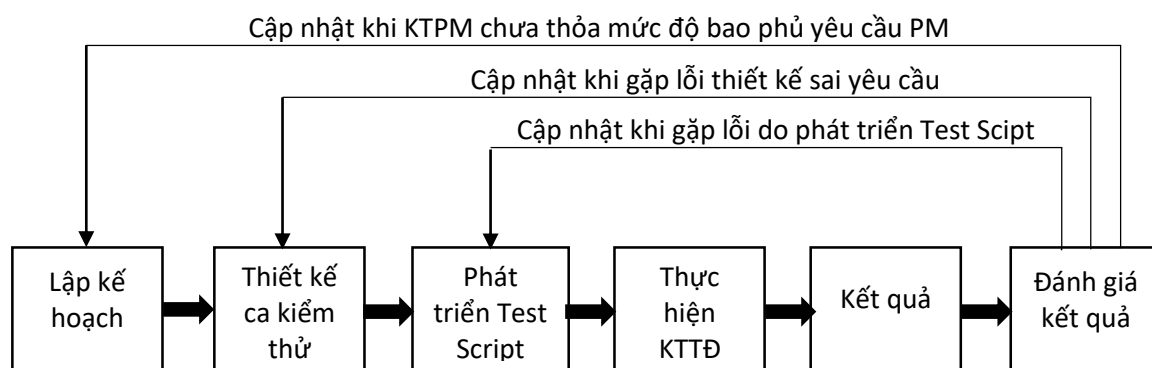
- Nếu test chương trình này cho cấp 1 thì chỉ cần có các chức năng cộng trừ nhân chia.
- Nếu test chương trình này cho cấp 2 thì phải thêm chức năng tính toán với đa thức/
- Nếu test chương trình này cho cấp 3 thì cần thêm tính toán tích phân, đạo hàm, v.v.

❖ *Nguyên tắc 7 – Sự sai lầm về việc không có lỗi*

Việc tìm và sửa chữa lỗi sẽ không giúp được gì nếu hệ thống được xây dựng xong nhưng không thể dùng được và không đáp ứng được nhu cầu và sự mong đợi của người dùng. (Nghĩa là nếu sau khi code, test rồi fix bug, làm đủ tất cả các trường hợp và cuối cùng cho ra một sản phẩm không như mong đợi hoặc không đáp ứng được nhu cầu của khách hàng thì dự án phần mềm đó coi như thất bại mặc dù đã được test xong).

2.4. Quy trình kiểm thử tự động

Quy trình kiểm thử tự động phần mềm cũng giống như quy trình thực hiện kiểm thử thủ công chỉ khác ở chỗ kiểm thử tự động có hỗ trợ của công cụ ít hoặc nhiều như tạo test script (có thể bằng tay hoặc công cụ), công cụ hỗ trợ về ghi lại kết quả và lưu trữ kết quả trong máy tính. Quy trình này cũng gần tương tự với quy trình phát triển phần mềm, được thực hiện qua nhiều bước, được tiến hành rất sớm trong quy trình phát triển phần mềm và đội kiểm thử tiến hành gần như song song cùng đội phát triển phần mềm.



Hình 2-1: Quy trình Kiểm thử tự động trong mối quan hệ với Kiểm thử phần mềm

Để kiểm thử tự động thì công cụ là thành phần không thể thiếu trong tiến trình này, việc kiểm thử viên thành thạo các công cụ kiểm thử đảm bảo cho quy trình kiểm thử tự động được hiệu quả.

2.5. Ưu điểm của kiểm thử tự động

- *Tiết kiệm tiền bạc và thời gian:* Nhận định này đặc biệt đúng nếu xét trong giai đoạn bảo trì của các dự án lớn. Mỗi tuần chúng ta phải thực hiện regression test từ 1 đến 2 lần với số lượng ca kiểm thử rất lớn trong 1 đến 2 ngày. Gần như không thể thực hiện cách thủ công, trong khi với kiểm thử tự động chúng ta hoàn toàn có thể với nguồn nhân lực vô cùng khiêm tốn.
- *Chính xác hơn:* Nhờ độ ổn định cao, kiểm thử tự động có thể thực thi các ca kiểm thử với độ chính xác cao hơn.
- *Độ bao phủ cao:* Như đã nói ở trên, khi sử dụng kiểm thử tự động, chúng ta có thể thực thi số lượng lớn ca kiểm thử trong một thời gian ngắn. Điều này giúp chúng ta tăng độ bao phủ trong giai đoạn kiểm thử hồi quy (một ví dụ điển hình).
- *Hoàn thành các công việc mà con người không thể làm được.*
- *Độ tin cậy cao:* Nhờ sự ổn định vượt trội của công cụ kiểm thử tự động so với con người, đặc biệt trong trường hợp có quá nhiều ca kiểm thử cần được thực thi, nên độ tin cậy của kiểm thử tự động thường cao hơn so với kiểm thử thủ công.

- *Khả năng lặp*: Hãy cùng xem xét một ví dụ: Trong một ngày thời tiết xấu chúng ta phải thực thi một ca kiểm thử với 50 bộ dữ liệu đầu vào khác nhau. Nếu thực thi cách thủ công, ngồi trước màn hình, nhập dữ liệu, click chuột, kiểm tra lại, v.v. trong 50 lần có lẽ bạn sẽ gục ngã sớm trên bàn làm việc của mình. Nhưng, nếu bạn thực thi bằng kiểm thử tự động, chỉ cần nhập dữ liệu vào file excel sau đó cho script chạy và ngồi làm việc khác cho tới khi nhận được báo cáo. Với độ ổn định cao, bạn hoàn toàn có thể tin tưởng vào kết quả thực thi của công cụ kiểm thử tự động.
- *Khả năng tái sử dụng*: Với một bộ kiểm thử tự động, chúng ta có thể sử dụng cho nhiều phiên bản ứng dụng khác nhau, đây được gọi là tính tái sử dụng.

2.6. Một số công cụ kiểm thử tự động

❖ Selenium

Selenium là một công cụ kiểm tra phần mềm được sử dụng để kiểm tra hồi quy. Đây là một công cụ kiểm tra mã nguồn mở cung cấp chức năng phát lại và thu âm để kiểm tra hồi quy. Các Selenium IDE chỉ hỗ trợ trình duyệt web Mozilla Firefox.

❖ QTP (HP UFT)

QTP được sử dụng rộng rãi để kiểm thử chức năng và hồi quy, giải quyết các ứng dụng phần mềm và môi trường. Để đơn giản hóa việc tạo và bảo trì thử nghiệm, nó sử dụng khái niệm kiểm tra từ khóa.

❖ Rational Function Tester

Là 1 công cụ kiểm tra tự động hướng đối tượng có khả năng tự động kiểm tra dữ liệu, kiểm tra giao diện, và kiểm thử hồi quy.

❖ WATIR

Là một phần mềm kiểm tra mã nguồn mở để kiểm thử hồi quy. Watir chỉ hỗ trợ khám phá Internet trên các cửa sổ trong khi Watir webdriver hỗ trợ Chrome, Firefox, IE, Opera, v.v.

❖ *Appium Studio*

Sẽ được đề cập đến trong phần thực nghiệm.

2.7. So sánh kiểm thử tự động và kiểm thử thủ công

Tiêu chí	Kiểm thử thủ công	Kiểm thử tự động
Thời gian	Mất nhiều thời gian thực thi nhưng không phải kiểm thử lặp đi lặp lại.	Mất ít thời gian thực thi nhưng quá trình kiểm thử lặp tăng hơn nhiều so với kiểm thử thủ công.
Độ linh động	Linh động do kiểm thử thủ công nên có thể phát hiện và xử lý những tình huống phát sinh trong quá trình kiểm thử. Và có thể tìm ra lỗi mới.	Không linh động vì kiểm thử theo script kiểm thử hiệu năng và tải trọng nên quá trình kiểm thử không phát hiện ra lỗi mới. Chỉ thích hợp với kiểm thử hồi quy.
Phụ thuộc	Phụ thuộc vào trạng thái của con người nên kết quả test có thể kém chính xác đối với dự án lớn có nhiều ca kiểm thử.	Nhất quán, nên kết quả kiểm thử là chính xác và không phụ thuộc vào yếu tố ngoại cảnh.
Bảo trì	Không cần bảo trì.	Cần bảo trì.
Kết quả	Có kết quả ngay lập tức.	Cần 1 thời gian mới có kết quả.
Ưu điểm	Kiểm thử linh hoạt và trong quá trình kiểm thử sẽ tìm đc ra lỗi mới.	Kiểm thử tự động thích hợp cho việc kiểm thử lặp đi lặp lại, có thể tái sử dụng testScript. Thích hợp giả lập kiểm thử hiệu năng, chịu tải cũng như giả lập hệ thống kiểm thử.
Hạn chế	Nếu sử dụng kiểm thử thủ công mà kiểm thử 1 chức năng lặp đi lặp lại thì sẽ tốn nhiều thời gian và sẽ khó chính xác. Nên thay thế bằng kiểm thử tự động để	Nếu sử dụng kiểm thử tự động mà kiểm thử ít sẽ rất lãng phí thời gian và nhân lực. Trong

	đỡ mất thời gian giám sát, tối ưu hóa việc sử dụng tài nguyên máy tính để kiểm thử.	trường hợp này thì nên thực hiện kiểm thử thủ công.
--	---	---

Bảng 2-2: So sánh kiểm thử tự động và kiểm thử thủ công

CHƯƠNG 3:

THỰC NGHIỆM SỬ DỤNG APPIUM STUDIO CHO KIỂM THỬ TỰ ĐỘNG TRÊN IOS

Chương cuối này sẽ là kết quả của quá trình thực nghiệm khi sử dụng một phần mềm điển hình cho việc kiểm thử tự động trên di động là Appium Studio. Nhưng do không có tài khoản của nhà phát triển trên IOS cũng như phiên bản miễn phí của phần mềm này bị giới hạn khả năng truy cập thiết bị trên đám mây, nên phải dùng một giải pháp khác đó là sử dụng một thành phần mở rộng Appium Studio có thể tích hợp vào trong phần mềm Eclipse Neon để thực hiện phần thực nghiệm này.

1. Giới thiệu phần mềm Appium Studio

1.1. Công cụ Appium

Appium là một công cụ mã nguồn mở được sử dụng để kiểm thử tự động (test automation) các native app, mobile web app, và hybrid app trên nền tảng iOS và Android.

Appium được phát triển từ nền tảng hệ thống Selenium (kế thừa các đối tượng, cấu trúc và cú pháp) nên Appium có khả năng làm việc với nhiều nền tảng khác nhau, iOS và Android, trên cùng một mã kiểm thử (cross-platforms).

Khả năng cross-platforms của Appium đến từ việc sử dụng các thư viện của Appium, thông qua một chương trình server, chuyển các câu lệnh sử dụng trong mã kiểm thử thành các lệnh UIAutomation (với iOS) hay UIAutomator (với Android) để tương tác với thiết bị.

Đặc biệt là Appium hỗ trợ “đa nền tảng” (cross-platform) cho phép bạn sử dụng API giống nhau để viết test cho các nền tảng khác nhau (iOS và Android). Điều này khá là tiện lợi khi bạn muốn sử dụng lại các bộ ca kiểm thử

của mình. Bên cạnh đó, Appium hỗ trợ viết test cho rất nhiều ngôn ngữ, từ Java cho đến Ruby, Python, JavaScript, v.v.

Các thành phần của Appium

- *Chương trình máy chủ Appium – Appium Server*: là một chương trình tạo lập một máy chủ Java, dùng để chuyển các lệnh trong mã kiểm thử thành các lệnh có thể tương tác với thiết bị: UIAutomation với iOS hay UIAutomator với Android.
- *Hệ thống thư viện Appium*: Cũng như Selenium, Appium có một hệ thống thư viện dùng để nhận diện và tương tác với các đối tượng UI trên ứng dụng di động. Hệ thống thư viện Appium được cung cấp cho nhiều ngôn ngữ lập trình khác nhau như C#, Java, Python, v.v. để kỹ sư kiểm thử tự động có thể chọn ngôn ngữ lập trình quen thuộc cho việc tự động hóa kiểm thử.

1.2. Phần mềm Appium Studio

Appium Studio là một IDE được thiết kế cho việc phát triển và thực thi kiểm thử tự động trên di động bằng cách sử dụng Appium/Selenium WebDriver API [10].

Phần mềm này loại bỏ phần lớn sự phụ thuộc cứng nhắc và các điều kiện tiên quyết của Appium chẳng hạn như yêu cầu phát triển của iOS trên các máy MAC OSX hoặc không có khả năng chạy các thử nghiệm song song trên các thiết bị iOS thực / mô phỏng. Giao diện của Appium Studio cung cấp một bộ công cụ để đơn giản hóa quá trình kiểm thử.

Những ưu điểm của Appium Studio:

- Có thể chạy kiểm thử trên hệ điều hành Windows.
- Đơn giản hoá việc thiết lập môi trường (không cần Xcode).
- Không bị giới hạn để kiểm tra ứng dụng của mình.
- Nhanh và đơn giản hóa quá trình kiểm thử.

Một số nhược điểm:

- Độ tùy biến không cao.
- Có phiên bản miễn phí nhưng không được hỗ trợ truy cập trên đám mây.

2. Appium Studio tích hợp trong Eclipse

Appium Studio dành cho Eclipse là một giải pháp dành cho các nhà phát triển Java muốn tự động kiểm thử ứng dụng trên thiết bị di động trên cả 2 nền tảng Android và iOS.

Appium Studio là một thành phần mở rộng để tự động hóa quá trình kiểm thử và có thể tích hợp vào JUnit, TestNG.

Các ưu điểm của Appium Studio dành cho Eclipse:

- Kết nối thiết bị đám mây và phản chiếu thiết bị: hỗ trợ truy cập từ xa vào thiết bị.
- Quản lý ứng dụng: Dễ dàng quản lý các ứng dụng di động của bạn.
- Có thể xác định đối tượng dễ dàng và cung cấp chế độ offline (phát triển các bài kiểm tra di động mà không internet hay thiết bị).
- Không cần kiểm thử viên phải hiểu được mã nguồn của chương trình.
- Kiểm thử một lúc được đa nền tảng và nhiều ca kiểm thử.

Một số nhược điểm:

- Không hỗ trợ kiểm thử trên thiết bị thật.
- Độ trễ của các thiết bị đám mây sẽ kéo dài thời gian kiểm thử.
- Đôi lúc hơi khó khăn để xác định đối tượng với những chương trình không rõ ràng.
- Bị giới hạn khả năng truy cập trên đám mây với bản miễn phí.
- Không thể kiểm thử được chính xác hiệu năng của phần mềm.
- Không thể kiểm thử khả năng gián đoạn khi có cuộc gọi, tin nhắn đến.

3. Thực nghiệm với Appium Studio tích hợp trong Eclipse

Phần thực nghiệm này sẽ kiểm tra các chức năng của một chương trình máy tính đơn giản chạy trên hệ điều hành IOS.

3.1. Cài đặt Appium Studio

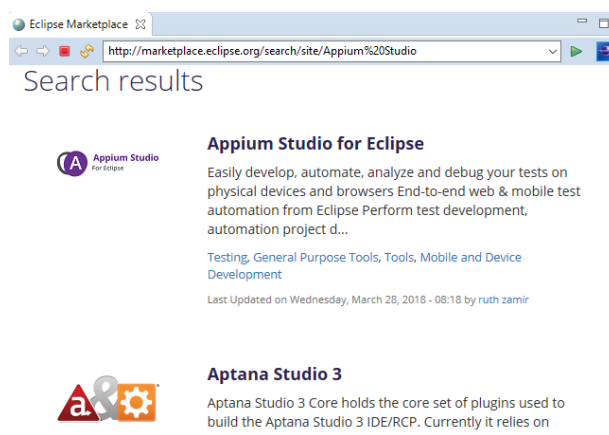
Bước 1: Truy cập địa chỉ:

http://www.eclipse.org/downloads/download.php?file=/technology/epp/download/release/neon/3/eclipse-jee-neon-3-win32-x86_64.zip để download và cài đặt.

Bước 2: Tiến hành download và cài đặt phần mềm Eclipse Neon

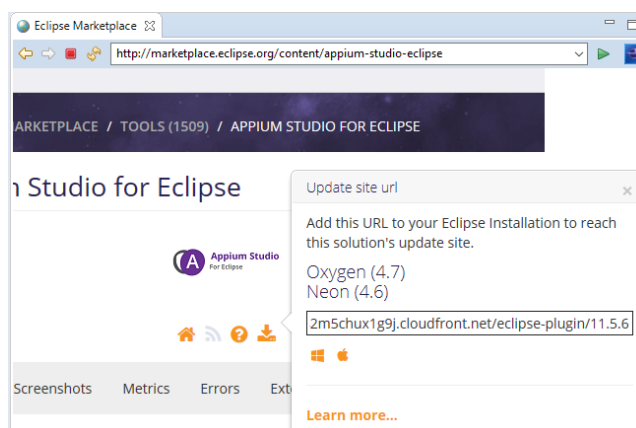
Bước 3: Khởi động phần mềm Eclipse Neon

Bước 4: Truy cập *Help/Eclipse Marketplace* và tìm kiếm từ khóa “Appium Studio”



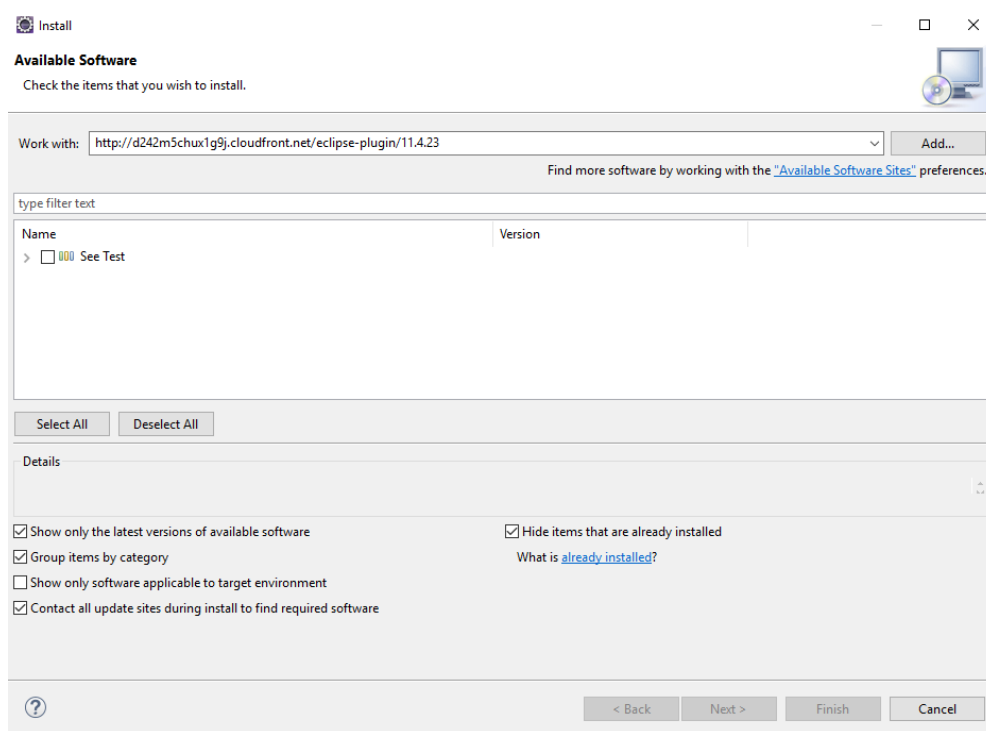
Hình 3-1: Kết quả tìm kiếm Appium Studio

Bước 5: Chọn kết quả trả về và lấy đường dẫn URL để cài đặt Appium Studio



Hình 3-2: Lấy URL để cài đặt Appium Studio

Bước 6: Truy cập *Help/Install New Software* sau đó dán URL vừa lấy được vào cửa sổ Install và tiến hành cài đặt.

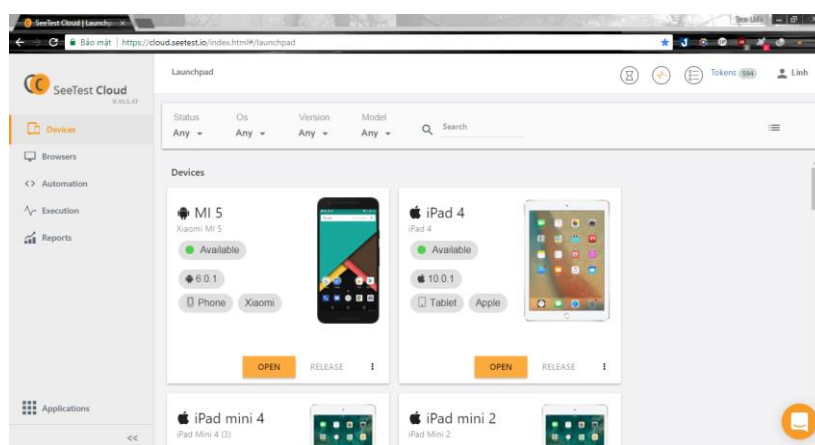


Hình 3-3: Dán URL vào cửa sổ Install để tiến hành cài đặt

Sau khi cài đặt, Eclipse sẽ yêu cầu khởi động lại → Tiến hành khởi động lại.

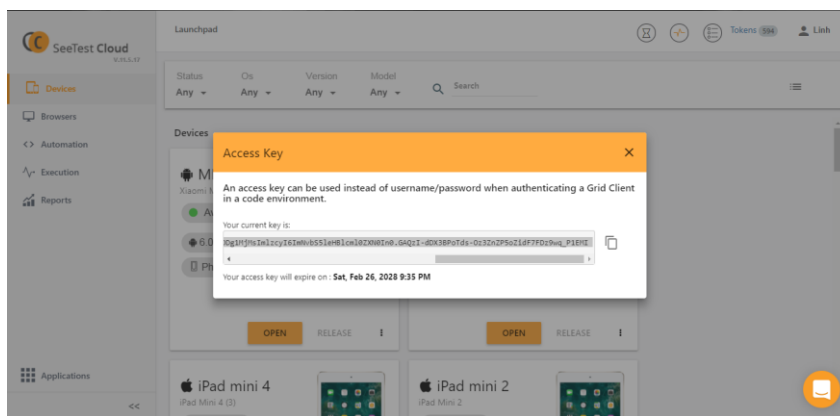
3.2. Kết nối với thiết bị trên Cloud

Bước 1: Truy cập <https://seetest.io/> rồi tiến hành đăng ký tài khoản và đăng nhập



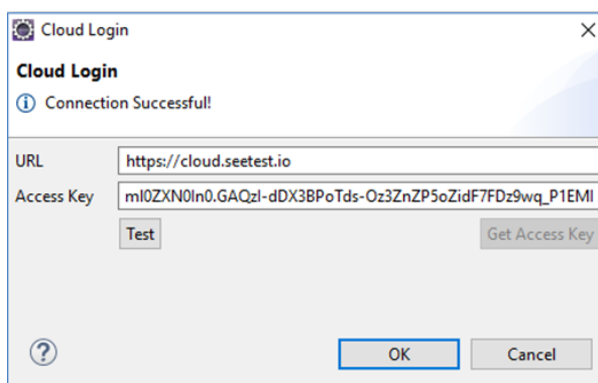
Hình 3-4: Giao diện trang Cloud của SeeTest

Bước 2: Nhấn vào tên ở góc phải rồi chọn “Get Access Key” sau đó sao chép lại



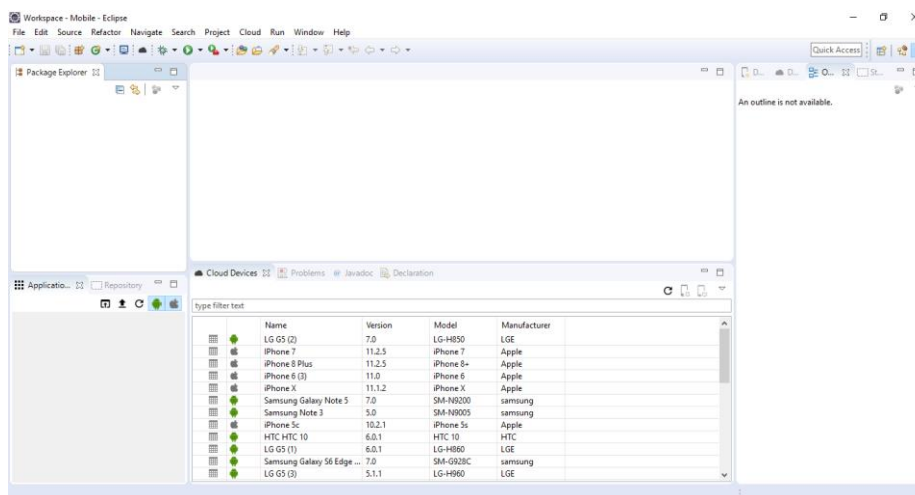
Hình 3-5: Copy lại Access Key

Bước 3: Mở Eclipse chọn Cloud/Cloud Connect. Ở thanh URL nhập <https://cloud.seetest.io> và thanh Access Key là Access Key vừa copy ở bước 2.

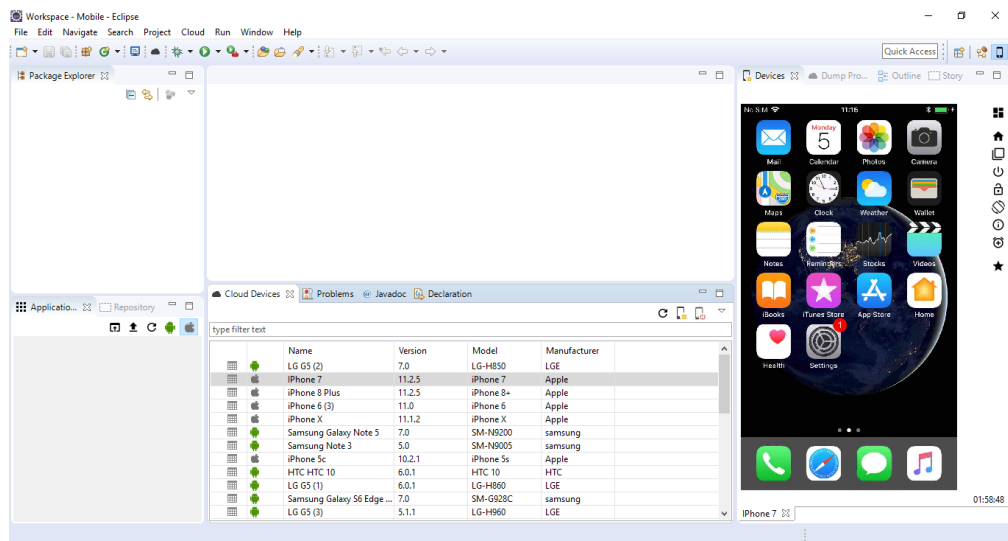


Hình 3-6: Kiểm tra kết nối đến máy chủ Cloud

Bước 4: Để nhìn thấy các thiết bị trên Cloud, ta phải mở thêm cửa sổ “Mobile” bằng cách chọn *Window/Perspective/Open perspective/Other* và chọn Mobile rồi nhấn OK.



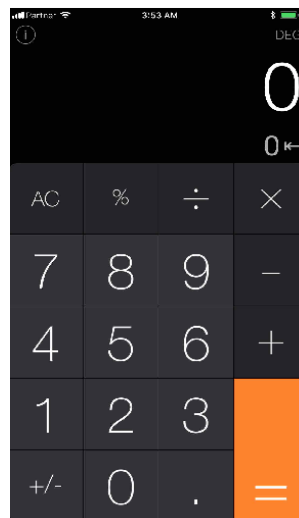
Hình 3-7: Các thiết bị Cloud được hiển thị trong Eclipse

Bước 5: Chọn 1 thiết bị để kết nối

Hình 3-8: Màn hình thiết bị được hiển thị sau khi kết nối

3.3. Xây dựng bộ ca kiểm thử cho một ứng dụng cần kiểm thử

Ở đây ta sẽ tiến hành kiểm thử cho một chương trình máy tính đơn giản và dùng kiểm thử hộp đen để kiểm tra các chức năng của phần mềm.



Hình 3-9: Giao diện chương trình máy tính cần kiểm thử

Yêu cầu của phần mềm:

- Có đầy đủ các chức năng +, -, *, /.
- Tính toán được với số tự nhiên và số thập phân.
- Tính toán được với số âm và số dương.
- Hiện kết quả ERROR với những lỗi toán học.

Áp dụng phương pháp phân vùng tương đương để xây dựng ca kiểm thử:

1. Xác định các lớp tương đương:

Với yêu cầu 2, 3 ta có

- Phân vùng 1, 2: Số tự nhiên âm, dương.
- Phân vùng 3, 4: Số thập phân âm, dương.
- Phân vùng 5: Số 0.

Với yêu cầu 4, áp dụng phương pháp kiểm thử biên với biên là số có 12 chữ số ta có thêm 3 trường hợp với số có 11, 12 và 13 chữ số.

Với yêu cầu 5, ta đã có các phép toán với phân vùng 5 (số 0).

2. Xây dựng các ca kiểm thử:

Ở đây ta sẽ chỉ kiểm thử với trường hợp những phép toán với 2 số và tạm bỏ qua phép tính với %.

Ta sẽ có số ca kiểm thử là:

$$(5 * 4)/2 * 3 * 4 = 120 \text{ (ca)}$$

Với $(5 * 4)/2$ là số ghép cặp các phân vùng, 3 là số ca kiểm thử với mỗi cặp (chưa tính phép toán và hoán vị trong phép toán, ví dụ cặp 1 và 2 sẽ có các ca kiểm thử với 11, 12 và 22) và 4 là số lượng phép toán (+, -, *, /).

Nếu kiểm thử bằng tay, sẽ rất khó khăn để kiểm thử được tất cả các trường hợp nêu trên. Ở đây, ta sẽ có một bộ các ca kiểm thử tiêu biểu như sau:

ID	Testcase Description	Test Data	Expected result
1	Kiểm thử phép cộng số nguyên âm với số thập phân dương.	-1 + 3.26	2.26
2	Kiểm thử phép trừ số nguyên dương với số thập phân âm.	9 - (-0.85)	9.85

3	Kiểm thử phép nhân số nguyên âm và số thập phân âm	$(-8)*(-5.17)$	41.36
4	Kiểm thử phép chia số thập phân dương cho số nguyên dương	$4.65 / 2$	2.325
5	Kiểm thử phép chia cho 0	$5 / 0$	ERROR

Hình 3-10: Bộ ca kiểm thử cho ứng dụng máy tính

Tuy nhiên ta sẽ không kiểm thử bằng tay mà sử dụng phần mềm để kiểm thử tự động. Phần mềm này tự sinh dữ liệu đầu vào theo yêu cầu, sau đó sẽ thực hiện tự động, cuối cùng là so sánh và đưa ra kết quả. Vẫn kiểm thử với các phép toán “+, -, *, /” với 2 số hạng bất kỳ.

3.4. Tạo dự án kiểm thử

3.4.1. Cài đặt ứng dụng

Bước 1: Chọn *File/New/Java Project* và đặt tên cho dự án rồi chọn Finish.

Bước 2: Chuột phải vào Project vừa tạo chọn *Configure/Mobile Nature*. Tiếp tục nhấn Finish.

```
package com.experitest.auto;

import java.net.URL;

import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Optional;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

import io.appium.java_client.ios.IOSDriver;
import io.appium.java_client.ios.IOSElement;
import io.appium.java_client.remote.IOSMobileCapabilityType;
import io.appium.java_client.remote.MobileCapabilityType;

public class IOSDemoTest extends BaseTest {
    protected IOSDriver<IOSElement> driver = null;

    @BeforeMethod
    @Parameters("deviceQuery")
    public void setUp(@Optional("@os='ios'") String deviceQuery) throws
Exception {
        init(deviceQuery);
        // Init application / device capabilities
    }
}
```

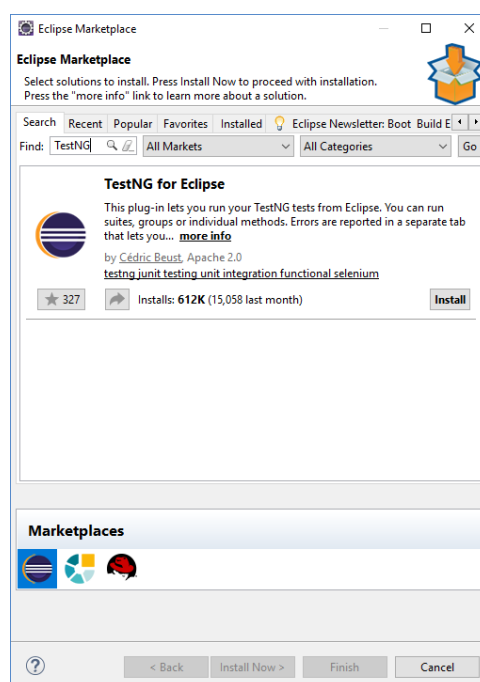
```
//dc.setCapability(MobileCapabilityType.APP,
"cloud:com.experitest.ExperiBank");
//dc.setCapability(IOSMobileCapabilityType.BUNDLE_ID,
"com.experitest.ExperiBank");
dc.setCapability("testName", "IOSDemoTest");
driver = new IOSDriver<>(new
URL(getProperty("url",cloudProperties) + "/wd/hub"), dc);
}

@Test
public void test() {
    // Enter the test code
}

@AfterMethod
public void tearDown() {
    driver.quit();
}
}
```

Hình 3-11: Đoạn code IOSTest được sinh tự động trong Project

Bước 3: Truy cập *Help/Eclipse Marketplace* và tìm kiếm từ khóa “TestNG” rồi tiến hành cài đặt



Hình 3-12: Kết quả tìm kiếm “TestNG”

Đây là một framework mã nguồn mở tự động; trong đó NG của TestNG có nghĩa là Next Generation - Thế hệ tiếp theo. TestNG tương tự như JUnit nhưng mạnh hơn Junit, nhưng nó vẫn được lấy cảm hứng từ JUnit.

Ưu điểm chính của TestNG là:

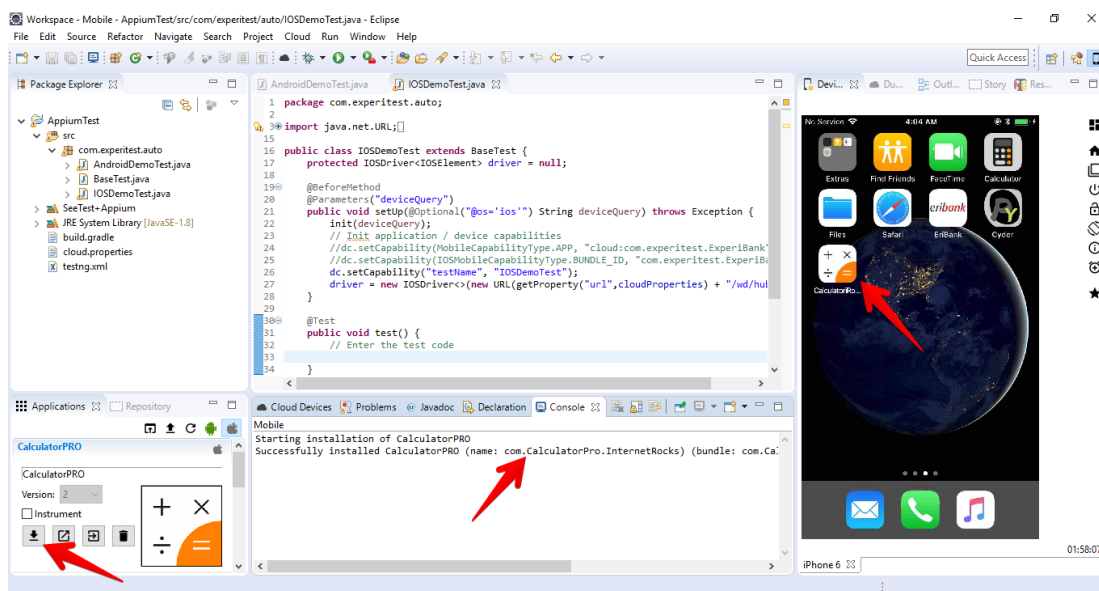
- Cho phép tạo ra các bản báo cáo HTML của tiến trình thực thi.
- Có các chú thích giúp việc kiểm thử dễ dàng hơn.
- Các trường hợp kiểm thử có thể được nhóm lại và được ưu tiên dễ dàng hơn.
- Có thể kiểm thử song song.
- Tạo ra các bản ghi.
- Có thể tham số hóa dữ liệu.

Sau khi cài đặt, Eclipse sẽ yêu cầu khởi động lại → Tiến hành khởi động lại.

Khi chạy Project, ta sẽ chạy bằng plug-in TestNG này.

Bước 4: Cài đặt ứng dụng cần kiểm thử lên thiết bị. Ở đây sẽ cài đặt 1 ứng dụng máy tính đơn giản.

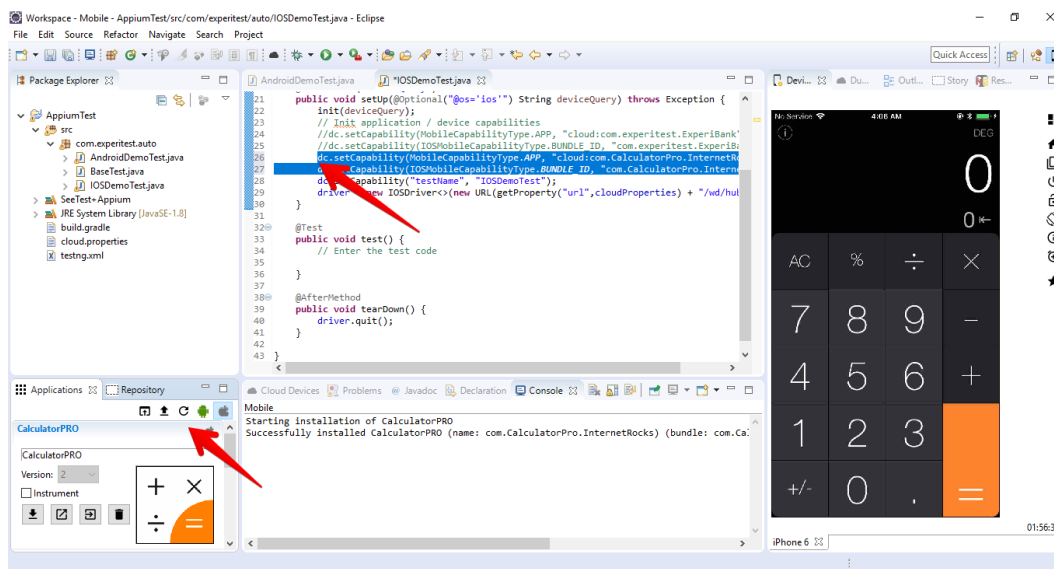
Đầu tiên ta cần upload ứng dụng lên cloud trước bằng cách chọn upload trong cửa sổ Applications góc trái bên dưới của Eclipse. Sau khi ứng dụng đã được upload, ta tiến hành cài đặt bằng nút Install cũng trong cửa sổ đó. Đợi 1 lúc sẽ có thông báo cài đặt thành công ở cửa sổ console.



Hình 3-13: Kết quả sau khi cài đặt ứng dụng Basic Calculator

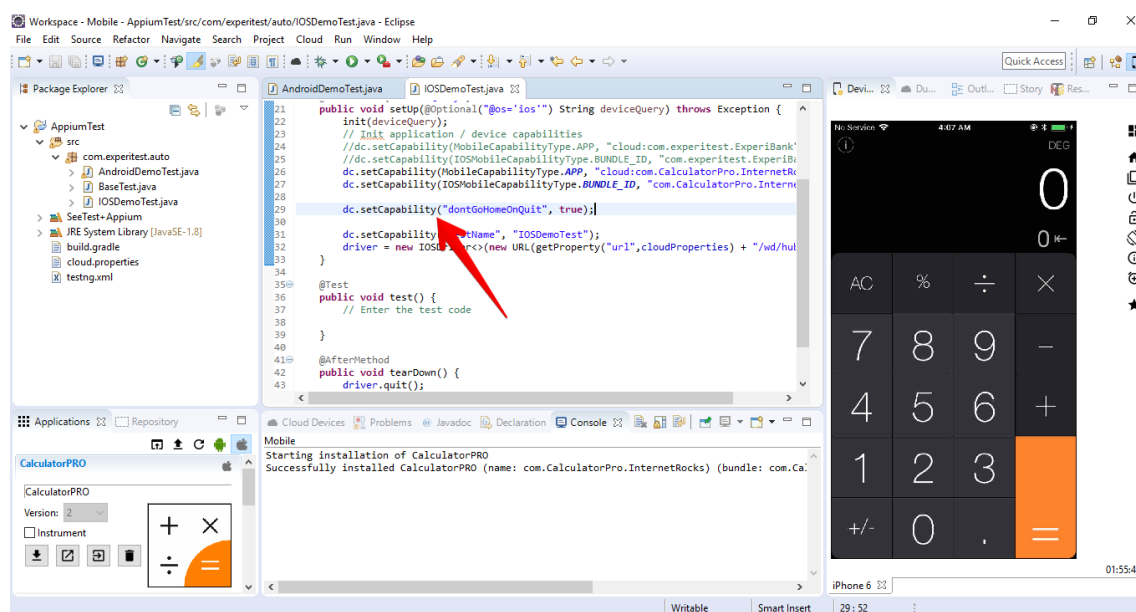
3.4.2. Viết kịch bản kiểm thử

Bước 1: Chuyển qua tab IOSDemoTest.java để tiến hành tạo kịch bản kiểm thử cho IOS. Đầu tiên sẽ là 2 dòng lệnh cài đặt ứng dụng, để chuột vào vùng bên phải cạnh tên ứng dụng ta vừa cài sau đó kéo và thả vào phần code setUp().



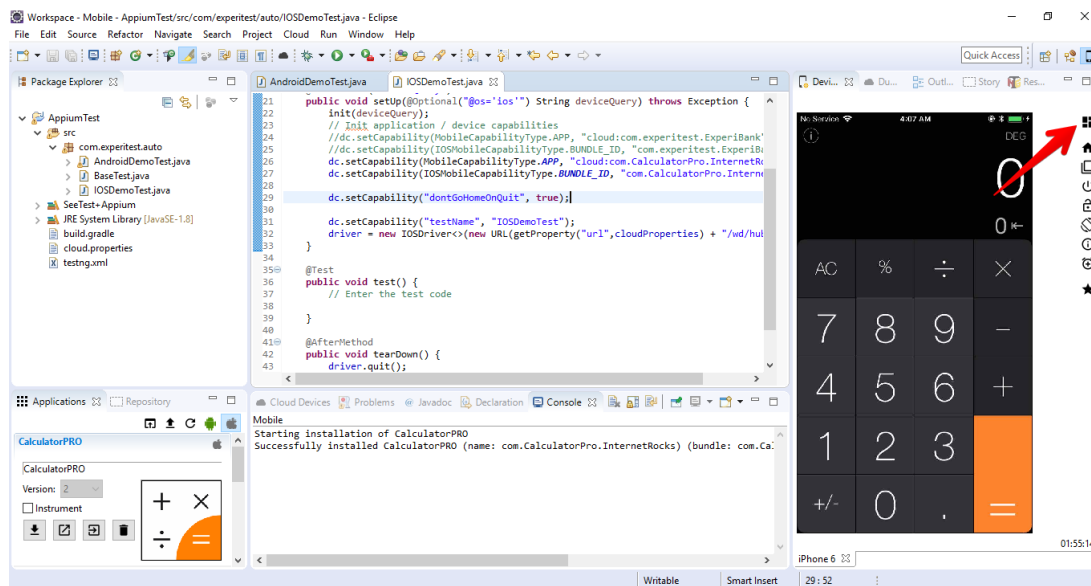
Hình 3-14: Code cài đặt ứng dụng được thêm vào phần setUp

Bước 2: Thêm dòng lệnh “dc.setCapability(“dontGoHomeOnQuit”, true);” để sau khi thực hiện test, ứng dụng sẽ không tự động thoát ra ngoài (có thể bỏ qua bước này).



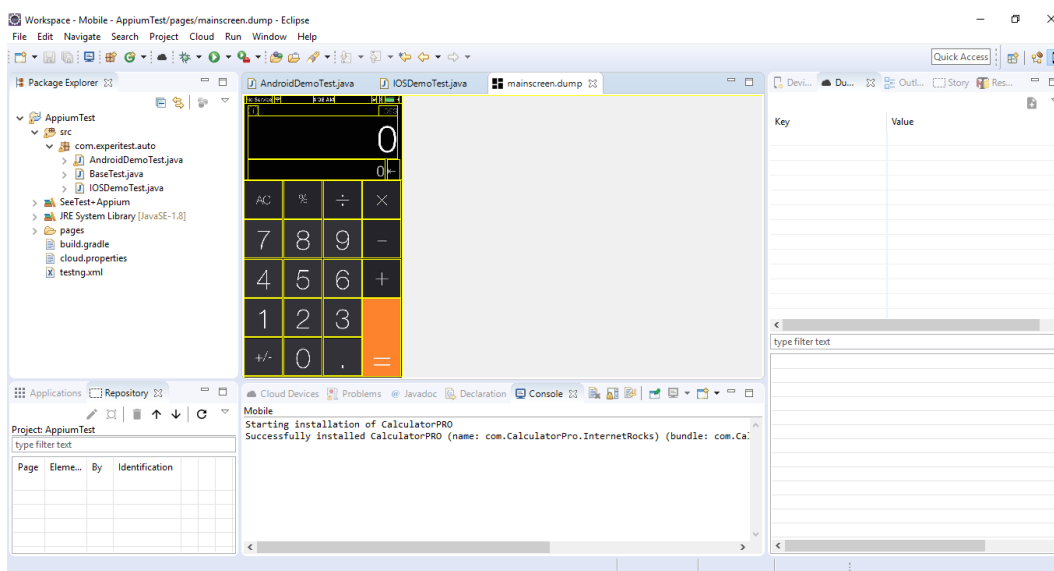
Hình 3-15: Thêm câu lệnh để chương trình không tự động thoát khi thực hiện kiểm thử

Bước 3: Chọn biểu tượng trên cùng của thanh menu phần cửa sổ hiển thị màn hình di động để mở cửa sổ Dump UI, màn hình này hiển thị đầy đủ các đối tượng UI để thuận tiện cho việc kiểm thử.



Hình 3-16: Chọn biểu tượng Dump UI ở cửa sổ Devices

Sau đó, ta lưu lại màn hình đó với tên tùy chọn, ở đây là “mainscreen.dump”.

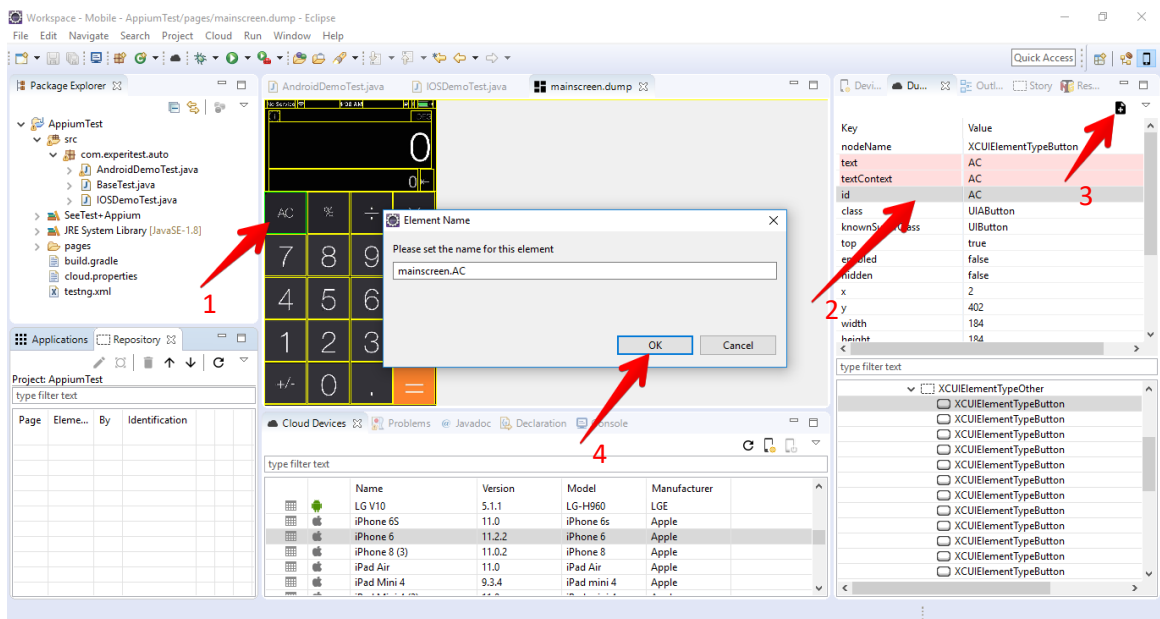


Hình 3-17: Màn hình được lưu với tên “mainscreen.dump”

Bước 4: Tiến hành lưu lại từng đối tượng trên màn hình.

Chọn từng đối tượng → Chọn một hoặc nhiều dòng Key (màu đỏ) ở cửa sổ Dump Properties (trường hợp không có dòng nào màu đỏ thì chọn một vài dòng có thể phân biệt với những đối tượng khác) → Chọn “Add to Repository”

ở góc trên bên phải → Nhập tên của đối tượng cần lưu (không được dùng ký tự đặc biệt).



Hình 3-18: Lưu lại đối tượng nút AC của màn hình máy tính

Bước 5: Viết code để sinh dữ liệu kiểm thử tự động

Tiến hành viết mã sinh số ngẫu nhiên trong phạm vi -999 → 999 (3 chữ số). Đầu tiên là đoạn mã sinh số thập phân ngẫu nhiên.

```

public float floatclick()
{
    int n;
    float a = 0, k = 100;
    Random rnd = new Random();

    //Nhập phần nguyên
    for (int i=0; i<3; i++)
    {
        n = rnd.nextInt(10);
        String str = "mainscreen." + Integer.toString(n);
        driver.findElement(in.Repo.obj(str)).click();
        a = a + (n * k);
        k = k/10;
    }

    driver.findElement(in.Repo.obj("mainscreen.point")).click();

    //Nhập phần thập phân
    k = (float)1/10;
    for (int i=0; i<3; i++)
    {
        n = rnd.nextInt(10);
        String str = "mainscreen." + Integer.toString(n);
        driver.findElement(in.Repo.obj(str)).click();
        a = a + (n * k);
    }
}
    
```

```

        k = k/10;
    }

    //Random so am duong (0 la am, 1 la duong)
    n = rnd.nextInt(2);

    //Kiem tra so am duong
    if (n == 0)
    {
        driver.findElement(in.Repo.obj("mainscreen.negative")).click();
        return -a;
    }
    else return a;
}

```

Hình 3-19: Đoạn mã sinh số thập phân ngẫu nhiên từ -999 đến 999

Với trường hợp sinh số nguyên ngẫu nhiên cũng tương tự nhưng đơn giản hơn:

```

public float intclick()
{
    int n;
    int a = 0, k = 100;
    Random rnd = new Random();

    //Nhap so nguyen
    for (int i=0; i<3; i++)
    {
        n = rnd.nextInt(10);
        String str = "mainscreen." + Integer.toString(n);
        driver.findElement(in.Repo.obj(str)).click();
        a = a + (n * k);
        k = k/10;
    }

    //Random so am duong
    n = rnd.nextInt(2);

    //Kiem tra so am duong
    if (n == 0)
    {
        driver.findElement(in.Repo.obj("mainscreen.negative")).click();
        return -a;
    }
    else return a;
}

```

Hình 3-20: Đoạn mã sinh số nguyên ngẫu nhiên từ -999 đến 999

Cuối cùng là đoạn mã chạy kiểm thử tự động. Vì để sinh dữ liệu kiểm thử tự động nên không thể bao phủ được hết mọi trường hợp, số lượng bộ dữ liệu đầu vào càng nhiều thì độ bao phủ càng lớn, các trường hợp lỗi càng dễ phát sinh. Phần này sẽ ví dụ trường hợp sinh ngẫu nhiên 20 bộ dữ liệu đầu vào.

```
public void test() {
    // Enter the test code
    //Khai bao 2 so hang
    Float a, b;

    Random rnd = new Random();
    Integer m, n;
    Float rs, kq;
    DecimalFormat df1 = new DecimalFormat();

    //Khai bao cac phep toan
    String math[] = {"plus", "minus", "multiply", "divide"};

    //Sinh 20 bo du lieu kiem thu ngau nhien
    for (int i=0; i<20; i++)
    {
        driver.findElement(in.Repo.obj("mainscreen.AC")).click();
        //Random so thu nhat (0 la so nguyen, 1 la so thap phan)
        n = rnd.nextInt(2);
        if (n==0) a = intclick();
        else a = floatclick();

        //Random phep toan
        rnd = new Random();
        m = rnd.nextInt(4);
        String mth = "mainscreen." + math[m];
        driver.findElement(in.Repo.obj(mth)).click();

        //Random so thu hai (0 la so nguyen, 1 la so thap phan)
        rnd = new Random();
        n = rnd.nextInt(2);
        if (n==0) b = intclick();
        else b = floatclick();

        //Ket qua
        driver.findElement(in.Repo.obj("mainscreen.equal")).click();

        //Lay ket qua hien thi tren man hinh
        String result =
driver.findElement(in.Repo.obj("mainscreen.result")).getText();
        String k = "";
        char res[] = result.toCharArray();
        int l = result.length();
        for (int j=0; j<l; j++)
            if (res[j] != ',') k = k + res[j];

        //kiem tra phep chia cho 0
        if ((m!=3) && (b!=0))
        {
            kq = Float.parseFloat(k);

            //Tinh toan ket qua thuc de so sanh
            if (m==0) rs = a + b;
            else if (m==1) rs = a - b;
            else if (m==2) rs = a * b;
            else rs = a / b;

            //So sanh ket qua test, Pass hay Fail
            if (df1.format(rs).equals(df1.format(kq)))
                System.out.print("PASS ");
            else System.out.print("FAIL ");
        }
    }
}
```



```

    }
    else
    {
        //So sanh ket qua test, Pass hay Fail
        if (k.equals("ERROR"))
            System.out.print("PASS ");
        else System.out.print("FAIL ");
    }

    //in ra phép tính và kết quả
    System.out.print(df1.format(a));
    if (m==0) System.out.print(" + ");
    else if (m==1) System.out.print(" - ");
    else if (m==2) System.out.print(" * ");
    else System.out.print(" / ");
    System.out.print(df1.format(b));
    System.out.print(" = ");
    System.out.print(result);
}
}

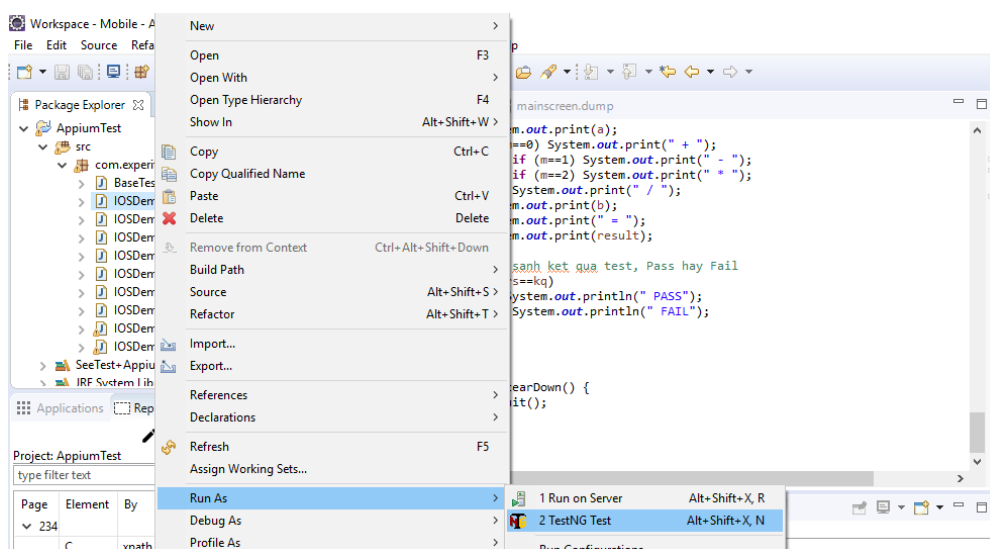
```

Hình 3-21: Đoạn mã sinh dữ liệu kiểm thử tự động

3.4.3. Khởi chạy kiểm thử tự động

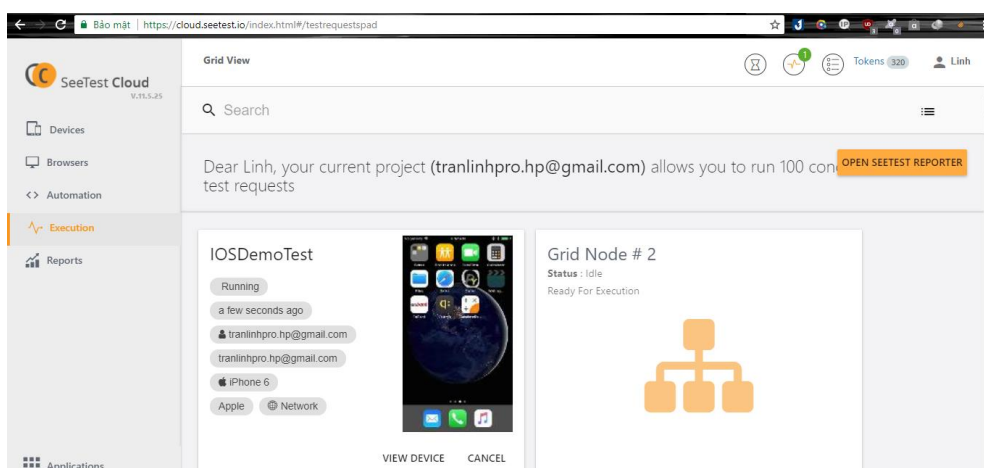
Chọn file java cần chạy rồi nhấn chuột phải → Run As → TestNG Test.

Phần này sẽ kiểm thử tự động trên thiết bị là Iphone 6 và hệ điều hành 11.2.2



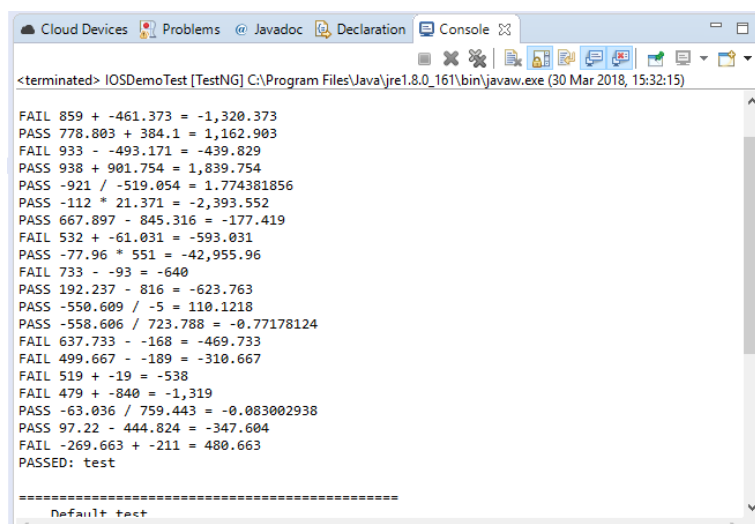
Hình 3-22: Khởi chạy kiểm thử tự động

Trong quá trình chạy test, ngoài xem trên thiết bị đang kết nối, ta có thể xem quá trình này trên web <https://cloud.seetest.io/>



Hình 3-23: Quá trình chạy kiểm thử trên web

Kết quả chạy kiểm thử tự động động sẽ được hiển thị trên cửa sổ console của Eclipse.



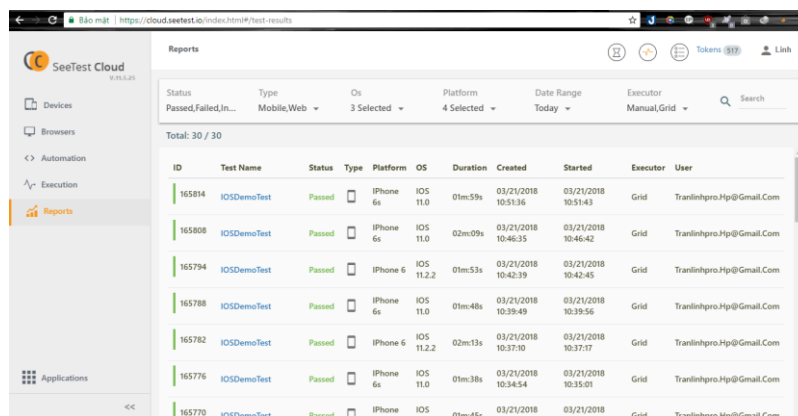
Hình 3-24: Kết quả sinh ca kiểm thử tự động

Có thể thấy khá nhiều bộ dữ liệu kiểm thử bị FAIL. Tất cả đều bị cùng một lỗi là khi số hạng thứ hai là số âm. Ta sẽ phải viết Bug Report cho trường hợp lỗi này ở phần sau.

3.5. Báo cáo

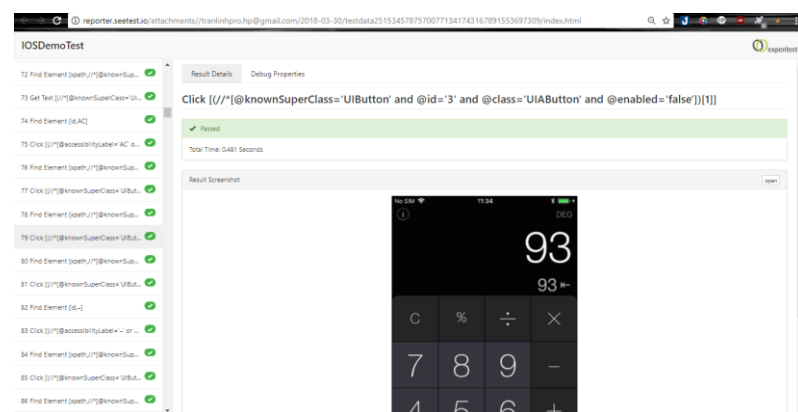
3.5.1. Xem báo cáo

SeeTest Cloud tự động sinh các báo cáo cho từng ca kiểm thử, giúp cho việc báo cáo của kiểm thử viên trở lên dễ dàng. Có rất nhiều thông tin được hiển thị như: thiết bị chạy test, phiên bản hệ điều hành, thời gian chạy test, v.v.



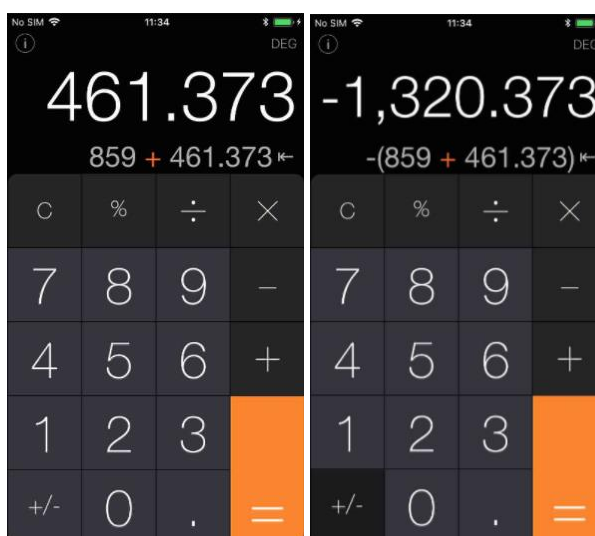
Hình 3-25: Toàn bộ báo cáo được sinh tự động trong phần Reports

Chọn một báo cáo để xem chi tiết. Từng bước thực hiện sẽ được chụp lại rất chi tiết giúp cho kiểm thử viên theo dõi được toàn bộ quá trình kiểm thử và dễ dàng nhận ra các lỗi phát sinh tiện lợi cho việc làm các báo cáo sau này.



Hình 3-26: Chi tiết quá trình thực hiện kiểm thử tự động

Có thể thấy ở bộ dữ liệu được sinh tự động đầu tiên đã phát sinh lỗi.



Hình 3-27: Ca kiểm thử đầu tiên không đưa ra kết quả chính xác

3.5.2. Tổng hợp báo cáo

Tổng hợp kết quả kiểm thử lại, ta được:

ID	Test Data	Expected Result	Actual Result	Result	Note
1	859 + -461.373	397.627	-1,320.373	FAIL	Button +/-
2	778.803 + 384.1	1,162.903	1,162.903	PASS	
3	933 - -493.171	1426.171	-439.829	FAIL	Button +/-
4	938 + 901.754	1,839.754	1,839.754	PASS	
5	-921 / -519.054	1.774381856	1.774381856	PASS	
6	-112 * 21.371	-2,393.552	-2,393.552	PASS	
7	667.897 - 845.316	-177.419	-177.419	PASS	
8	532 + -61.031	470.969	-593.031	FAIL	Button +/-
9	-77.96 * 551	-42,955.96	-42,955.96	PASS	
10	733 - -93	640	-640	FAIL	Button +/-
11	192.237 - 816	-623.763	-623.763	PASS	
12	-550.609 / -5	110.1218	110.1218	PASS	
13	-558.606 / 723.788	-0.77178124	-0.77178124	PASS	
14	637.733 - -168	805.733	-469.733	FAIL	Button +/-
15	499.667 - -189	688.667	-310.667	FAIL	Button +/-
16	519 + -19	500	-538	FAIL	Button +/-
17	479 + -840	-361	-1,319	FAIL	Button +/-
18	-63.036 / 759.443	-0.083002938	-0.083002938	PASS	
19	97.22 - 444.824	-347.604	-347.604	PASS	
20	-269.663 + -211	-480.663	480.663	FAIL	Button +/-

Hình 3-28: Tổng hợp kết quả kiểm thử


3.5.3. Viết Bug report

BUG REPORTS

Project: CalculatorPro

Reported by: Tran Linh, Bui

- **Bug Name:** The (+/-) button is not working correctly
- **Bug ID:** Cal0001
- **Date:** 30-Mar-2018
- **Assigned to:** Impala Studios
- **Status:** New
- **Summary/Description:**
The (+/-) button is not working correctly with the second number in a calculation.
- **Environments (OS/Browser):** Iphone 6 – IOS 11.2.2
- **Step to reproduce:**
 1. Input the first number.
 2. Select the operation minus (-), plus (+), etc.
 3. Input the second number.
 4. Press the (+/-) and the bug will appear. My expected result is “859 + (-461.373)” but I have “-(859 + 461.373)”.
- **Actual results:** The TextBox display the result “-1,320.373”.
- **Expected results:** The TextBox display the result “397.627”.
- **Severity:** Major (S2)
- **Priority:** High (P1)
- **Attachment:**
<http://reporter.seetest.io/attachments//tranlinhpro.hp@gmail.com/2018-03-30/testdata25153457875700771341743167891553697309/index.html>



Hình 3-29: Bug report lỗi của nút (+/-)

KẾT LUẬN

Sau một thời gian tìm hiểu và nghiên cứu đề tài này, em đã đạt được một số kết quả sau:

- Tìm hiểu được về một pha rất quan trọng và không thể thiếu trong quá trình phát triển phần mềm, đó là kiểm thử. Đặc biệt đi sâu hơn trong kiểm thử trên thiết bị di động.
- Tìm hiểu được cách xây dựng ca kiểm thử, tạo ra được Bug report.
- Ứng dụng thành công phần mềm Appium Studio tích hợp trong Eclipse để sinh dữ liệu đầu vào cho kiểm thử tự động chức năng của một phần mềm trên hệ điều hành IOS.

Tuy nhiên vẫn còn tồn tại một số vấn đề và hạn chế:

- Không thể kiểm thử trên thiết bị di động thật do không có tài khoản Apple Developer.
- Vì phải kiểm thử bằng thiết bị trên đám mây nên đôi khi độ trễ lớn dẫn đến các ca kiểm thử bị kéo dài và không thể kiểm tra được chính xác hiệu năng của chương trình khi chạy.
- Phần mềm Appium Studio không còn hỗ trợ người dùng miễn phí truy cập đến đám mây nên đã phải chuyển sang kiểm thử qua phần mềm Eclipse.
- Appium Studio còn khá mới và chưa có nhiều tài liệu hướng dẫn về phần mềm này.
- Chưa kiểm thử được về mặt giao diện, hiệu năng, sự hao tốn pin, khả năng khả năng kết nối, tính tương thích, v.v.
- Phần mềm máy tính chạy trên IOS để kiểm thử chưa cho thấy rõ được đặc trưng của các ứng dụng chạy trên nền tảng này.

Em rất mong nhận được sự góp ý của các Thầy, Cô và các bạn để có thêm kinh nghiệm và kiến thức để tiếp tục nghiên cứu và hoàn thiện nội dung nghiên cứu trong đề tài này.

TÀI LIỆU THAM KHẢO

- [1] Nguyễn Văn Hà, Nguyễn Văn Vy, Giáo trình Kỹ nghệ phần mềm, Hà Nội: Nhà xuất bản giáo dục, 2009.
- [2] C. Kaner, Exploratory Testing, Orlando, FL: Florida Institute of Technology, November 2006.
- [3] IEEE, Thuật ngữ Công nghệ Phần Mềm.
- [4] Trương Anh Hoàng, Đặng Văn Hưng, Phạm Ngọc Hùng, Giáo trình kiểm thử phần mềm, Hà Nội: Đại học Quốc gia Hà Nội, Tháng 1 năm 2014.
- [5] <https://labs.septeni-technology.jp>, SQA.
- [6] Wikipedia, Bách khoa toàn thư mở.
- [7] <https://viblo.asia>, QA.
- [8] <http://softwaretestingfundamentals.com>, Defect Severity & Priority.
- [9] <http://istqbexamcertification.com>, ISTQB EXAM CERTIFICATION.
- [10] <https://docs.experitest.com>, Experitest Self-Training.