

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA: CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN SEMINAR**  
**Đề tài: Automation testing for mobile apps**  
**MÔN: KIỂM THỬ PHẦN MỀM**

**NHÓM THỰC HIỆN – NHÓM F2:**

**MSSV: 20120049 – HỌ TÊN: Nguyễn Hải Đăng**

**MSSV: 20120084 – HỌ TÊN: Nguyễn Văn Hiếu**

**MSSV: 20120458 – HỌ TÊN: Hồ Sĩ Đức**

**MSSV: 20120467 – HỌ TÊN: Nguyễn Phước Hải**

**Giảng viên hướng dẫn: ThS. Trần Duy Hoàng**

**Giảng viên thực hành: ThS. Trương Phước Lộc**

**Lớp lý thuyết: 20\_1**

**Học kỳ - Niên khoá: HK1 - 2023-2024**

# MỤC LỤC

<b>I. Danh sách thành viên – Phân công đồ án seminar.....</b>	<b>3</b>
<b>II. Lý thuyết .....</b>	<b>3</b>
1. Khái niệm automation testing và mobile automation testing. ....	3
a. Automation testing: .....	3
b. Mobile automation testing.....	3
2. Quy trình của mobile automation testing. ....	4
3. Lợi ích và nhược điểm.....	5
a. Lợi ích .....	5
b. Nhược điểm.....	6
4. Những trường hợp nào nên dùng Automation Testing cho ứng dụng Mobile?.....	6
5. So sánh các công cụ mobile automation testing .....	7
a. Katalon .....	7
b. Appium.....	8
c. TestComplete.....	9
d. Ranorex Studio.....	11
e. Nightwatch.js .....	13
f. Espresso.....	15
g. Quantum.....	17
h. Kobiton.....	18
i. Robotium .....	19
j. AcceLQ.....	19
6. Bảng so sánh các công cụ kiểm thử.....	21

## I. Danh sách thành viên – Phân công đồ án seminar

Họ và tên	Mã số sinh viên	Phân công
Nguyễn Hải Đăng	20120049	Viết tài liệu + Slide mục 1, 3, 5c, 5d
Nguyễn Văn Hiếu	20120084	Viết tài liệu + Slide mục 2, 4, 5h, 5j
Hồ Sĩ Đức	20120458	Viết tài liệu + Slide mục 5a, 5g, 5i
Nguyễn Phước Hải	20120467	Viết tài liệu + Slide mục 5b, 5e, 5f

## II. Lý thuyết

### 1. Khái niệm automation testing và mobile automation testing.

#### *a. Automation testing:*

Kiểm thử tự động là việc thực hiện kiểm thử bằng máy mà người kiểm thử sẽ khởi động hệ thống, nhập dữ liệu đầu vào, kiểm tra và so sánh với dữ liệu đầu ra và ghi lại kết quả.

Những kiểm thử này đóng vai trò cực kỳ quan trọng trong việc giảm thiểu sai sót, nâng cao năng suất kiểm thử và giảm sự nhầm lẫn của việc kiểm thử thủ công lặp đi lặp lại trong một thời gian dài.

Automation testing là quá trình tự động xử lý các bước để thực hiện một kịch bản kiểm thử được thực hiện bởi phần mềm Automation testing tool.

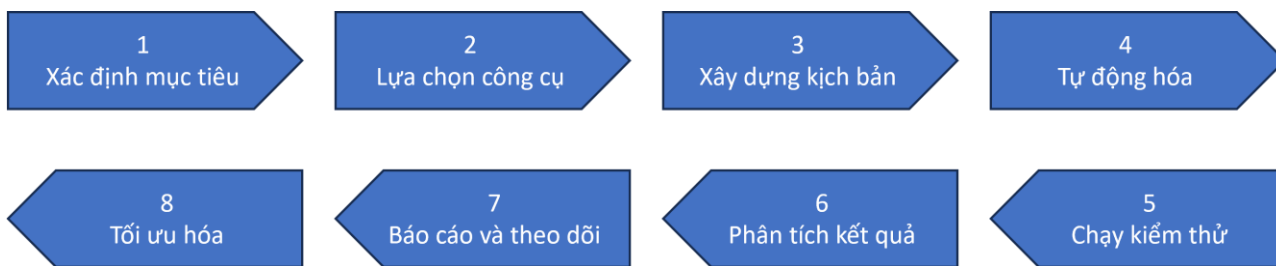
Mục đích của các thử nghiệm này là phát hiện lỗi với mục tiêu cuối cùng là tạo ra sản phẩm có chất lượng tốt nhất.

#### *b. Mobile automation testing*

- Mobile testing là quá trình kiểm tra chức năng, hiệu suất, khả năng sử dụng và tính nhất quán trên các thiết bị di động như điện thoại di động, máy tính bảng. Mục tiêu chính của mobile testing là đảm bảo rằng ứng dụng hoạt động đúng và tương thích trên các thiết bị khác, nền tảng, hệ điều hành khác nhau và trong điều kiện sử dụng khác nhau.

- Mobile testing được chia làm 2 loại là:
  - Kiểm thử thủ công (manual testing): là quá trình kiểm thử hoàn toàn bằng con người, tester sẽ sử dụng các chiến lược, giải pháp để tìm ra và ghi nhận lại vấn đề mà ứng dụng di động gặp phải. Tất cả quá trình trên đều được thực hiện thủ công.
  - Kiểm thử tự động (automation testing): là quá trình kiểm thử bỏ qua một số nhiệm vụ nhất định khỏi con người bằng cách sử dụng các công cụ kiểm thử (tools) và kịch bản tự động, giải phóng tester tập trung vào các nhiệm vụ kiểm thử khác.

## 2. Quy trình của mobile automation testing.



Quy trình kiểm thử:

1. **Xác định mục tiêu kiểm thử:** Đây là bước đầu tiên trong quy trình, nơi chúng ta xác định các mục tiêu cụ thể mà chúng ta muốn đạt được thông qua quá trình kiểm thử tự động. Mục tiêu bao gồm các tính năng cần được kiểm thử và các kịch bản kiểm thử cần thiết.
2. **Lựa chọn công cụ kiểm thử:** Chọn công cụ kiểm thử phù hợp với nhu cầu kiểm thử cụ thể của ứng dụng di động.
3. **Xây dựng kịch bản kiểm thử:** Tạo các kịch bản kiểm thử dựa trên yêu cầu và tính năng của ứng dụng di động. Đảm bảo rằng các kịch bản kiểm thử đảm bảo phủ đầy đủ các tính năng chính và các trường hợp sử dụng quan trọng.

**4. Tự động hóa kiểm thử:** Sử dụng công cụ kiểm thử đã chọn để tự động hóa các kịch bản kiểm thử đã xác định. Viết mã tự động hóa phù hợp để thực hiện các thao tác kiểm thử trên ứng dụng di động.

**5. Chạy kiểm thử tự động:** Thực hiện các kịch bản kiểm thử tự động trên các thiết bị di động và máy ảo để kiểm tra tính đúng đắn và hiệu suất của ứng dụng.

**6. Phân tích kết quả kiểm thử:** Phân tích kết quả kiểm thử để xác định xem các tính năng nào đang hoạt động đúng và nếu có lỗi nào đang xảy ra.

**7. Báo cáo và theo dõi:** Tạo báo cáo chi tiết về kết quả kiểm thử, bao gồm các lỗi được phát hiện, cải thiện hiệu suất, và các phản hồi về chất lượng tổng thể của ứng dụng. Theo dõi việc sửa lỗi và cải thiện chất lượng sau mỗi chu kỳ kiểm thử.

**8. Tối ưu hóa quy trình:** Dựa trên kết quả kiểm thử và phản hồi từ người dùng, tối ưu hóa quy trình kiểm thử tự động để đảm bảo rằng nó liên tục cải thiện chất lượng của ứng dụng di động.

### 3. Lợi ích và nhược điểm.

#### a. Lợi ích

- Tốc độ: Kiểm thử tự động có thể chạy nhanh hơn nhiều so với kiểm thử thủ công, đặc biệt khi có nhiều trường hợp kiểm thử.
- Tính lặp lại: Kiểm thử tự động có thể chạy nhiều test-case với cùng một quy trình, đảm bảo tính nhất quán trong kiểm thử.
- Độ tin cậy: kiểm thử tự động ít xảy ra lỗi của con người hơn, giúp kết quả đáng tin cậy hơn.
- Tiết kiệm thời gian: Kiểm tra tự động có thể tiết kiệm thời gian về lâu dài vì chúng không yêu cầu nỗ lực thủ công cho các tác vụ lặp đi lặp lại.
- Tích hợp liên tục: Kiểm thử tự động có thể được tích hợp vào quy trình tích hợp liên tục, cho phép phát hiện sớm các lỗi.

### ***b. Nhược điểm***

- Chi phí thiết lập ban đầu: Kiểm thử tự động đòi hỏi chi phí để thiết lập và duy trì, bao gồm đầu tư vào công cụ, tập lệnh và cơ sở hạ tầng.
- Chi phí bảo trì: Kiểm thử tự động cần được cập nhật và duy trì khi công cụ có sự thay đổi, dẫn đến chi phí bảo trì bổ sung.
- Độ phức tạp: Kiểm tra tự động có thể phức tạp và khó hiểu, đặc biệt đối với các bên liên quan phi kỹ thuật.
- Kết quả “pass” giả: Kiểm tra tự động có thể bỏ sót một số lỗi, đặc biệt là những lỗi liên quan đến trải nghiệm người dùng.

## **4. Những trường hợp nào nên dùng Automation Testing cho ứng dụng Mobile?**

- Kiểm thử tự động cho các kịch bản kiểm thử lặp đi lặp lại: Khi có các kịch bản kiểm thử mà cần được thực hiện nhiều lần, automation testing có thể giúp tiết kiệm thời gian và tăng tính chính xác.
- Kiểm thử tích hợp: Khi cần kiểm tra sự tương tác giữa ứng dụng di động và các thành phần hệ thống bên ngoài, automation testing có thể giúp đảm bảo rằng tích hợp diễn ra một cách đúng đắn.
- Kiểm thử đa nền tảng và đa thiết bị: Khi ứng dụng cần hoạt động trên nhiều loại thiết bị và nền tảng khác nhau, automation testing có thể giúp đảm bảo tính nhất quán và tính đa dạng của ứng dụng trên các nền tảng này.
- Kiểm thử hiệu suất và độ ổn định: Khi cần đánh giá hiệu suất và độ ổn định của ứng dụng dưới tải, automation testing có thể tạo ra tải tự động và đo lường hiệu suất theo cách thức được thiết lập trước đó.
- Kiểm thử đa ngôn ngữ: Khi ứng dụng hỗ trợ nhiều ngôn ngữ, automation testing có thể giúp đảm bảo rằng tất cả các phiên bản ngôn ngữ của ứng dụng hoạt động đúng như mong đợi trên mọi thiết bị và hệ điều hành.
- Kiểm thử tương thích thiết bị và hệ điều hành: Automation testing có thể giúp kiểm tra tính tương thích của ứng dụng trên nhiều loại thiết bị di động và phiên bản hệ điều hành khác nhau.

## 5. So sánh các công cụ mobile automation testing.

### a. Katalon

- Link website: <https://katalon.com/>
- Mô tả: Katalon Studio là một công cụ kiểm thử tự động và tự động hóa thử nghiệm phần mềm được phát triển bởi Katalon, Inc. Nó giúp các nhà phát triển và nhóm kiểm thử thực hiện kiểm tra tự động cho ứng dụng web, ứng dụng di động và dịch vụ web. Katalon Studio cung cấp một môi trường tích hợp để tạo, quản lý và chạy các kịch bản kiểm tra tự động một cách dễ dàng mà không cần biết nhiều về lập trình.
- Danh sách chức năng:
  - Giao diện dựa trên GUI: Katalon Studio cung cấp một giao diện đồ họa dễ sử dụng để tạo và quản lý các kịch bản kiểm tra, cho phép người dùng không cần có kiến thức chuyên sâu về lập trình.
  - Hỗ trợ nhiều nền tảng: Nó hỗ trợ kiểm tra ứng dụng web, ứng dụng di động trên nhiều hệ điều hành như iOS và Android, và dịch vụ web.
  - Báo cáo và lưu trữ kết quả: Katalon cung cấp báo cáo chi tiết về kết quả kiểm tra và lưu trữ lịch sử kiểm tra.
  - Hỗ trợ Kiểm thử đám mây ( Katalon TestCloud ).
  - Ghi video và phát lại: Cung cấp khả năng ghi và phát lại cho việc tạo kịch bản kiểm thử một cách thuận tiện.
  - Khả năng tích hợp: Tích hợp mạch lạc với Git, Jenkins, qTest và Jira, nâng cao khả năng hợp tác và tích hợp liên tục.
- Điểm mạnh:
  - Cài đặt và sử dụng đơn giản
  - Dễ tiếp cận đối với những bạn mới bắt đầu học automation
  - Hỗ trợ cho người có chuyên môn về lập trình: Đối với những người có chuyên môn về lập trình, Katalon cung cấp chế độ tạo kịch bản trong chế độ kịch bản bằng code.
  - Tạo bài kiểm thử hiệu quả thông qua các template sẵn có



- Kiểm thử đa trình duyệt dễ dàng
- Báo cáo kiểm thử tự động , trực quan: Báo cáo kết quả kiểm thử có giao diện trực quan và có thể được xuất ra định dạng PDF và CSV.
- 
- Điểm yếu:
  - Framework mới được phát triển nên cộng đồng support vẫn còn hạn chế
  - Chỉ hỗ trợ lập trình trên ngôn ngữ Groovy và Java
  - Performance hơi chậm khi thực hiện automation với những project lớn khoảng vài nghìn test case
  - Dùng bản free nên sẽ bị limit 1 vài chức năng. Nếu muốn dùng đầy đủ tính năng thì phải mua license

#### ***b. Appium***

- Link website: <https://appium.io/docs/en/2.1/>
- Mô tả: Appium là một công cụ mã nguồn mở được sử dụng để kiểm thử tự động trên các ứng dụng di động trên các nền tảng iOS và Android. Appium được xây dựng trên cơ sở của WebDriver, một tiêu chuẩn tự động hóa trình duyệt web. Với kiến trúc đa nền tảng, Appium cho phép viết các kịch bản kiểm thử một lần và chạy chúng trên cả iOS và Android mà không cần sửa đổi mã nguồn.
- Danh sách chức năng:
  - Hỗ trợ đa nền tảng: Appium cung cấp một API thống nhất để tương tác với các thiết bị di động trên các nền tảng iOS và Android.
  - Hỗ trợ nhiều ngôn ngữ lập trình: Appium hỗ trợ nhiều ngôn ngữ lập trình như Java, Python, Ruby, JavaScript và C#.
  - Hỗ trợ các ứng dụng native, hybrid và web.
  - Tìm kiếm các thành phần giao diện đa nền tảng.
  - Hỗ trợ kiểm thử trên thiết bị thật và giả lập (Emulators và Simulators).
  - Hỗ trợ các cử chỉ ngón tay: Appium cung cấp các cử chỉ ngón tay phổ biến trên di động như: chạm, vuốt, cuộn, pinch và zoom.



- Tích hợp nhiều Framework kiểm thử: Appium tích hợp các Framework phổ biến như TestNG, JUnit và XCTest.
- Hỗ trợ Kiểm thử trên đám mây (Cloud testing).
- Điểm mạnh:
  - Là một công cụ miễn phí và mã nguồn mở.
  - Hỗ trợ đa nền tảng.
  - Hỗ trợ nhiều ngôn ngữ lập trình.
- Điểm yếu:
  - Cấu hình và quản lý khó khăn
  - Tốc độ chậm: việc sử dụng giao thức JSON Wire Protocol và WebDriver làm cho Appium chậm hơn một số công cụ kiểm thử khác.
  - Thiết độ chính xác so với các công cụ kiểm thử gốc (native testing tools).

### c. *TestComplete*

- Link website: <https://smartbear.com/product/testcomplete/overview/>
- Mô tả: TestComplete có thể tự động hóa việc kiểm tra giao diện người dùng chức năng cho các ứng dụng máy tính để bàn, thiết bị di động và web. Với sự hỗ trợ tích hợp cho hơn 500 cách điều khiển và framework của bên thứ ba, TestComplete có thể xử lý và xác định các thành phần giao diện người dùng động trong hầu hết các công nghệ hiện có.
- Danh sách chức năng:
  - Ngôn ngữ lập trình được hỗ trợ: JavaScript, Python, VBScript, JScript, Delphi, C++, C#.
  - Các phương pháp linh hoạt để thiết kế kiểm thử: chế độ ghi và phát lại, chế độ thủ công và tập lệnh với các từ khóa tích hợp.
  - Nhận dạng đối tượng bằng nhận dạng hình ảnh dựa trên thuộc tính và AI.
  - Kiểm thử song song, đa trình duyệt và đa thiết bị.
  - Tích hợp với các khung kiểm thử, công cụ CI/CD khác và hệ sinh thái SmartBear.

- **Điểm mạnh:**

- **Tính linh hoạt:** TestComplete hỗ trợ nhiều công nghệ, bao gồm các ứng dụng web, thiết bị di động, máy tính để bàn và ứng dụng lai. Nó có thể được sử dụng để thử nghiệm các ứng dụng được xây dựng bằng các ngôn ngữ và khung lập trình khác nhau.
- **Kiểm thử đa nền tảng:** TestComplete cho phép bạn thực hiện kiểm thử tự động trên nhiều hệ điều hành khác nhau như Windows, macOS, iOS và Android. Tính linh hoạt này rất hữu ích khi nhắm mục tiêu nhiều nền tảng.
- **Nhận dạng đối tượng và viết kịch bản:** TestComplete cung cấp khả năng nhận dạng đối tượng mạnh mẽ, giúp xác định và tương tác với các thành phần khác nhau trong ứng dụng dễ dàng hơn. Ngôn ngữ kịch bản của TestComplete dựa trên JavaScript, cho phép tùy chỉnh và tự động hóa thử nghiệm mạnh mẽ.
- **Ghi và phát lại:** TestComplete cung cấp tính năng ghi và phát lại, cho phép người kiểm tra ghi lại các tương tác của họ với một ứng dụng và tự động tạo tập lệnh kiểm thử. Tính năng này đơn giản hóa quá trình tạo thử nghiệm ban đầu.
- **Bộ thư viện kiểm thử và tích hợp phong phú:** TestComplete bao gồm một bộ thư viện kiểm tra toàn diện và tích hợp tích hợp với các công cụ và khung phát triển phổ biến. Nó tích hợp tốt với hệ thống CI/CD, hệ thống kiểm soát phiên bản, công cụ theo dõi lỗi, v.v.
- **Trực quan hóa kiểm tra:** TestComplete cung cấp nhật ký trực quan và kết quả thực hiện kiểm tra, giúp phân tích và gỡ lỗi các lỗi kiểm tra dễ dàng hơn. Nó cung cấp các báo cáo chi tiết kèm theo ảnh chụp màn hình và thông tin về phạm vi kiểm tra, hỗ trợ phân tích kiểm tra.

- **Điểm yếu:**

- **Chi phí:** TestComplete là một công cụ thương mại và chi phí bản quyền có thể là rào cản đối với một số tổ chức, đặc biệt là các nhóm nhỏ hơn hoặc những người thử nghiệm cá nhân – Chi phí bản quyền khá đắt đỏ, khoảng 3655 đô la (theo website của SmartBear).

- Chỉ hỗ trợ Windows: Mặc dù có thể thực hiện các kiểm thử tự động trên các nền tảng khác nhau nhưng bản thân công cụ này chỉ hoạt động trên Windows và không hỗ trợ sử dụng trên các máy Linux hoặc Mac vốn ngày càng trở nên phổ biến hơn trong thế giới phát triển phần mềm.
- Chi phí bảo trì: Giống như bất kỳ công cụ kiểm tra tự động nào có nhiều tính năng tích hợp, TestComplete yêu cầu bảo trì thường xuyên để tránh các thử nghiệm tùy thuộc vào những tính năng này bị hỏng. Khi các ứng dụng phát triển và thay đổi, các tập lệnh kiểm thử có thể cần cập nhật, điều này có thể làm tăng chi phí bảo trì tổng thể.
- Tùy chọn kiểm thử đám mây hạn chế: Mặc dù TestComplete cho phép thử nghiệm đa nền tảng, nhưng các tùy chọn kiểm tra đám mây của nó bị hạn chế hơn so với một số công cụ khác. Nếu kiểm thử của bạn yêu cầu cơ sở hạ tầng dựa trên đám mây rộng rãi, bạn có thể cần xem xét các công cụ bổ sung.

#### **d. Ranorex Studio**

- Link website: <https://www.ranorex.com/>
- Mô tả: Ranorex Studio là một giải pháp tự động hóa kiểm thử phần mềm mạnh mẽ dành cho kiểm thử ứng dụng máy tính, ứng dụng di động và trang web. Nó cung cấp một môi trường đồ họa để tạo và quản lý các kịch bản kiểm thử tự động và hỗ trợ việc kiểm tra trên nhiều nền tảng.
- Danh sách chức năng:
  - Ghi và phát lại kịch bản kiểm thử tự động: Ranorex Studio cho phép người dùng ghi các hoạt động trên ứng dụng hoặc trang web và sau đó phát lại chúng để kiểm tra.
  - Hỗ trợ kiểm thử đa nền tảng: Ranorex Studio hỗ trợ kiểm thử trên nhiều nền tảng như Windows, web, Android, iOS và nhiều hệ điều hành khác.
  - Quản lý kiểm thử: Nó cho phép quản lý dự án kiểm thử, tổ chức và thực hiện các ca kiểm thử, và tạo báo cáo chi tiết về kết quả kiểm thử.

- Hỗ trợ cho ngôn ngữ lập trình: Ranorex hỗ trợ sử dụng mã nguồn mở để tùy chỉnh kịch bản kiểm thử bằng cách sử dụng C# hoặc VB.NET.
- Hỗ trợ cho kiểm thử liên tục: Nó có tích hợp với các công cụ quản lý phiên bản và dịch vụ liên tục như Jenkins, TeamCity, và Git.
- Điểm mạnh:
  - Kiểm thử tự động đa nền tảng: Với Ranorex, bạn có thể chạy thử nghiệm tự động cho các ứng dụng web và desktop cũng như ứng dụng dành cho thiết bị di động, trong khi các sản phẩm cạnh tranh như Selenium và Katalon Studio không hỗ trợ thử nghiệm trên ứng dụng desktop và Watir chỉ nhằm mục đích kiểm thử web.
  - Tạo test-case mà không cần code: Ranorex Recorder với giao diện kéo và thả cho phép thực hiện các test-case không cần code. Do đó, các bài kiểm tra rất dễ đọc và thiết kế nhanh chóng vì các từ khóa có thể được sử dụng lại trong các trường hợp kiểm thử.
  - Sử dụng dễ dàng: Với giao diện người dùng thân thiện với người dùng, Ranorex được coi là một trong những công cụ tự động hóa kiểm thử dễ tiếp cận nhất trên thị trường.
  - Theo dõi image: Tính năng nhận dạng đối tượng thông minh trong Ranorex Studio tự động phát hiện bất kỳ thay đổi nào về vị trí của image trong GUI đã được kiểm thử trong giao diện của nó.
- Điểm yếu:
  - Bản quyền trả phí: Ranorex là một công cụ có trả phí, đây có thể là một nhược điểm đáng kể khi xét đến nhiều lựa chọn thay thế cạnh tranh nguồn mở như Selenium, Katalon Studio và Watir.
  - Chỉ có một số ngôn ngữ được hỗ trợ: Mặc dù Ranorex cung cấp tính linh hoạt về nền tảng và trình duyệt nhưng nó chỉ hỗ trợ hai ngôn ngữ kịch bản: C# và VB.NET, không giống như Selenium cho phép mã hóa bằng mười ngôn ngữ lập trình phổ biến.
  - MacOS không thể cài Ranorex Studio.

- Cộng đồng sử dụng không lớn, chính vì vậy nếu có bug sẽ khó trong việc tìm hiểu.
- Các phiên bản của Ranorex Studio không mang tính ổn định, thường xuyên được cập nhật => người dùng sẽ mất thời gian để cập nhật hơn.

#### ***e. Nightwatch.js***

- Link website: <https://nightwatchjs.org/>
- Mô tả: Nightwatch.js là một framework linh hoạt cho kiểm thử tự động mà mở rộng khả năng của mình để bao gồm kiểm thử cho ứng dụng di động, cả trên nền tảng Android và iOS, thông qua việc tích hợp với Appium.
- Danh sách chức năng:
  - Tích hợp với Appium: Tích hợp mượt mà với Appium cho phép kiểm thử tự động của ứng dụng di động native trên cả Android và iOS.
  - Khung kiểm thử thống nhất: Cung cấp một khung thử nghiệm thống nhất để viết kiểm thử bằng JavaScript, đảm bảo sự nhất quán giữa các kịch bản kiểm thử trên web và di động.
  - Kiểm thử chức năng Từ Đầu Đến Cuối: Hỗ trợ kiểm thử chức năng toàn diện của ứng dụng di động native, bao gồm tương tác người dùng và xác minh tính năng ứng dụng.
  - Kiểm thử chéo nền tảng: Cho phép kiểm thử trên nhiều nền tảng di động khác nhau, đảm bảo hiệu suất nhất quán trên các hệ điều hành khác nhau.
  - Ngôn ngữ JavaScript: Sử dụng JavaScript làm ngôn ngữ lập trình, mang lại sự dễ tiếp cận và sự quen thuộc cho nhà phát triển viết kiểm thử di động.
  - Thực hiện kiểm thử đồng thời: Hỗ trợ thực hiện kiểm thử đồng thời cho ứng dụng di động, tăng cường hiệu suất kiểm thử và giảm thời gian thực hiện kiểm thử tổng thể.
- Điểm mạnh:

- Hiệu suất chéo nền tảng: Hỗ trợ chéo nền tảng mượt mà với Appium để kiểm thử thống nhất trên cả iOS và Android, giảm cần thiết phải có các kịch bản riêng biệt.
- Khung kiểm thử thống nhất: Nightwatch.js tối ưu hóa nỗ lực kiểm thử với một khung thử nghiệm thống nhất cho cả ứng dụng web và di động, giảm độ dốc học.
- Tích hợp lệnh toàn diện: Bộ lệnh tích hợp sẵn trong Nightwatch.js giúp đơn giản hóa việc tạo kịch bản kiểm thử, cho phép mô phỏng tương tác người dùng và kiểm thử các điều kiện.
- Dễ sử dụng: Nổi tiếng với sự đơn giản, Nightwatch.js, kết hợp với Appium, mang lại trải nghiệm thân thiện với người dùng thông qua cú pháp rõ ràng.
- Ưu tiên JavaScript: Tận dụng JavaScript, Nightwatch.js tích hợp mượt mà với hệ sinh thái rộng lớn của nó, nâng cao khả năng tự động hóa kiểm thử.
- Thực hiện kiểm thử đồng thời: Nightwatch.js hỗ trợ thực hiện kiểm thử đồng thời, giảm thời gian kiểm thử bằng cách chạy nhiều kiểm thử cùng một lúc trên nhiều thiết bị hoặc giả lập.
- Điểm yếu:
  - Hiệu năng: Đôi khi, Nightwatch.js có thể gặp vấn đề về hiệu năng khi thực hiện kiểm thử đồng thời trên nhiều thiết bị, đặc biệt là với các ứng dụng lớn và phức tạp.
  - Quản lý trạng thái: Quản lý trạng thái của ứng dụng trong quá trình kiểm thử có thể là một thách thức, đặc biệt là khi cần duy trì một trạng thái cụ thể giữa các bước kiểm thử.
  - Hạn chế trong xử lý ảnh: Nightwatch.js có thể gặp khó khăn khi xử lý các kịch bản kiểm thử liên quan đến xác nhận và so sánh hình ảnh.
  - Cộng đồng và tài nguyên hạn chế: So với một số công cụ khác, cộng đồng Nightwatch.js có thể không lớn như mong đợi, điều này có thể ảnh hưởng đến việc tìm kiếm hỗ trợ và tài nguyên trực tuyến.



- Khả năng mở rộng và tích hợp: Trong một số trường hợp, việc mở rộng và tích hợp Nightwatch.js với môi trường kiểm thử tổ chức có thể đòi hỏi nhiều công sức hơn so với một số công cụ khác.
- Chưa có sự hỗ trợ mạnh mẽ cho kiểm thử đa ngôn ngữ.

#### ***f. Espresso***

- Link website: <https://developer.android.com/training/testing/espresso>
- Mô tả: Espresso là một framework được cung cấp và phát triển bởi google nhằm phục vụ cho việc kiểm thử giao diện trên nền tảng Android.
- Danh sách chức năng:
  - Khả năng đồng bộ hóa tự động: Espresso đảm bảo giao diện người dùng hoặc ứng dụng đã khởi động trước khi kiểm thử bắt đầu, với khả năng đồng bộ hóa tự động.
  - Android Intents: Quản lý hành vi của Android Intents, giúp kiểm thử trở nên mạnh mẽ và giảm sự phụ thuộc vào sự thay đổi của ứng dụng bên ngoài.
  - Tự động đồng bộ hóa với giao diện người dùng: Espresso tự động đồng bộ hóa với các phần tử và hành động kiểm thử trong giao diện người dùng.
  - Tính linh hoạt và mở rộng cao: Espresso có API đơn giản, nhẹ, dễ học và dễ bảo trì, với khả năng linh hoạt và mở rộng cao.
  - Hỗ trợ kiểm thử unit và black-box
  - Module Riêng Biệt Cho Kiểm Thử Android WebView và Android Intents:
  - Hỗ trợ lấy phản hồi nhanh chóng: Cho phép nhà phát triển nhận phản hồi nhanh chóng về thay đổi mã và sửa lỗi.
  - Không sử dụng máy chủ: Espresso không phụ thuộc vào máy chủ như Selenium RemoteWebDriver, chạy cùng với ứng dụng và cung cấp kết quả ngay lập tức.
- Điểm mạnh:
  - Đồng bộ tự động: Espresso cung cấp chức năng đồng bộ tự động cho các thành phần giao diện người dùng, phát hiện thời điểm không hoạt động của luồng



chính để thực hiện kiểm thử một cách ổn định và nhanh chóng mà không cần kết nối đến máy chủ.

- Dễ sử dụng: Được ưa chuộng bởi nhà phát triển Android, đặc biệt là với Espresso Test Recorder, nó giúp đơn giản hóa quá trình tạo kiểm thử giao diện người dùng mà không cần kỹ năng lập trình cao. Tích hợp dễ dàng với Android Studio, giúp việc gỡ lỗi và phân tích kết quả kiểm thử trở nên thuận tiện.
- Luồng làm việc đơn giản và phản hồi nhanh chóng: Espresso biên dịch bộ kiểm thử tự động thành các tệp APK, chạy chúng cùng với ứng dụng trên thiết bị để đạt được phản hồi nhanh chóng mà không cần kết nối đến máy chủ.
- API ngắn gọn: Với một API đơn giản bao gồm viewMatchers, viewActions và viewAssertions, Espresso cung cấp phản hồi nhanh chóng nhờ vào khả năng độc lập với kết nối đến máy chủ.

- Điểm yếu:

- Hạn chế về ngôn ngữ lập trình: So với Appium, Espresso hỗ trợ ít ngôn ngữ hơn, chỉ bao gồm Java và Kotlin. Điều này làm cho nó trở thành công cụ ưa chuộng chủ yếu cho những nhà phát triển ứng dụng Android native. Tuy nhiên, có hạn chế khi ngăn chặn ngôn ngữ sử dụng.
- Không hỗ trợ chéo nền tảng: Mặc dù Espresso có nhiều ưu điểm, nhưng khả năng không hỗ trợ kiểm thử chéo nền tảng. Lựa chọn Espresso hạn chế đội ngũ kiểm thử chỉ thực hiện kiểm thử trên Android, điều này có thể gây bất tiện khi tổ chức muốn mở rộng ứng dụng trên nhiều nền tảng. Đòi hỏi họ phải tìm kiếm một framework tương tự cho kiểm thử giao diện người dùng trên nền tảng iOS, dẫn đến công việc kép.
- Tập trung chủ yếu vào kiểm thử giao diện người dùng: Espresso có xu hướng tập trung nhiều vào kiểm thử giao diện người dùng, và do đó, không phải là lựa chọn tốt nhất cho các ứng dụng hoặc phần mềm có nhiều logic nghiệp vụ hoặc thành phần không phải giao diện. Bên cạnh đó, Espresso không cung cấp sự hỗ trợ rộng rãi cho việc kiểm thử phân tán, điều này có thể là một hạn chế đối với các dự án lớn hoặc phức tạp đòi hỏi sự phối hợp giữa nhiều đội và môi trường.

- Hơn nữa, việc tự động hóa với Espresso yêu cầu bạn phải có quyền truy cập vào mã nguồn của ứng dụng, điều này có thể làm cho quá trình tự động hóa kiểm thử trở nên phức tạp.

#### **g. Quantum**

- Link website: <https://www.perfecto.io/integrations/quantum>
- Mô tả: Quantum là một framework kiểm thử tự động mã nguồn mở được phát triển bởi Perfecto. Nó cung cấp khả năng kiểm thử tự động trên cả ứng dụng di động và trình duyệt web. Quantum hỗ trợ kiểm thử trên các nền tảng di động như iOS và Android.
- Danh sách chức năng:
  - Kiểm thử đa nền tảng: Quantum hỗ trợ kiểm thử trên cả hai nền tảng di động chính là iOS và Android.
  - Xử lý sự kiện người dùng trên thiết bị di động: Quantum cho phép bạn mô phỏng các sự kiện người dùng như chạm, vuốt, kéo và thả trên màn hình của ứng dụng di động.
  - Chạy kiểm thử đa thiết bị: Quantum hỗ trợ chạy các kịch bản kiểm thử trên nhiều thiết bị di động cùng một lúc. Bạn có thể kiểm tra ứng dụng trên các thiết bị iOS và Android khác nhau, trên các thiết bị có kích thước màn hình khác nhau.
  - Tạo báo cáo kết quả kiểm thử: Quantum cung cấp các báo cáo chi tiết về kết quả kiểm thử, giúp theo dõi và phân tích kết quả kiểm thử một cách dễ dàng.
  - Tích hợp với các công cụ kiểm thử liên tục (CI/CD): Quantum có thể tích hợp vào quy trình CI/CD của bạn, cho phép bạn thực hiện kiểm thử tự động và liên tục trong quá trình phát triển ứng dụng di động.
- Điểm mạnh:
  - Đơn giản và dễ sử dụng: Quantum sử dụng cú pháp dễ đọc và viết, gần gũi với ngôn ngữ tự nhiên.
  - Tích hợp với các công cụ kiểm thử liên tục (CI/CD).
  - Tương thích với Appium.
- Điểm yếu:

- Là một công cụ mới nên độ ổn định và khả năng sử dụng thua kém so với các một số công cụ trước đó.
- Ít tài liệu và cộng đồng hỗ trợ phát triển.
- Không cung cấp đầy đủ các tính năng và chức năng.

#### ***h. Kobiton***

- Link website: <https://kobiton.com/>
- Mô tả: Kobiton là một nền tảng kiểm thử và triển khai ứng dụng di động dựa trên cloud. Nó cung cấp các công cụ và dịch vụ để kiểm thử ứng dụng di động trên nhiều thiết bị và nền tảng khác nhau.
- Danh sách chức năng:
  - Kiểm thử đa nền tảng: Kobiton cho phép kiểm thử ứng dụng di động trên nhiều hệ điều hành, bao gồm iOS và Android.
  - Quản lý thiết bị từ xa thông qua cloud.
  - Tích hợp các công cụ kiểm thử tự động phổ biến như Appium và Selenium.
  - Hỗ trợ kiểm thử thu công.
  - Hỗ trợ quản lý vòng đời ứng dụng.
  - Kobiton giúp quản lý quá trình phát triển và triển khai ứng dụng di động trên các thiết bị thật và máy ảo. Kobiton hỗ trợ phát hành, triển khai và giám sát ứng dụng của mình.
  - Cung cấp các báo cáo và phân tích chi tiết về quá trình kiểm thử.
- Điểm mạnh:
  - Quản lý thiết bị dễ dàng: Kobiton quản lý và kiểm soát các thiết bị qua cloud cho phép người dùng truy cập và thao tác từ xa giúp cho việc quản lý dễ dàng.
  - Tiết kiệm thời gian.
  - Hỗ trợ kiểm thử đa nền tảng.
  - Tích hợp các công cụ kiểm thử tự động phổ biến như Appium và Selenium.
- Điểm yếu:
  - Là một công cụ trả phí.

### *i. Robotium*

- Link website: <https://github.com/RobotiumTech/robotium>
- Mô tả: Robotium là một framework kiểm thử mã nguồn mở được tạo ra để giúp việc viết các trường hợp kiểm thử giao diện người dùng (UI) tự động mạnh mẽ và chắc chắn cho ứng dụng di động Android.
- Danh sách chức năng:
  - Kiểm thử trên các ứng dụng Android bao gồm ứng dụng native và hybrid.
  - Cần yêu cầu tối thiểu kiến thức về ứng dụng được kiểm thử (application under test).
  - Có thể làm việc, xử lý nhiều activity trong Android một cách tự động.
  - Các testcase tin cậy hơn do liên kết (ràng buộc) run-time với các thành phần giao diện người dùng.
  - Tương thích với nhiều phiên bản Android.
  - Tích hợp các công cụ phổ biến như Maven, Gradle và Ant.
- Điểm mạnh:
  - Là công cụ mã nguồn mở và miễn phí
  - Dễ sử dụng.
  - Thời gian cần thiết để viết các ca kiểm thử (testcase) là rất ít.
  - Thời gian thực thi kiểm thử tự động nhanh chóng.
  - Tích hợp, tương thích tốt với Maven, Gradle và Ant.
  - Làm việc, xử lý tự động nhiều activity trong Android.
  - Tương thích với nhiều phiên bản Android.
- Điểm yếu:
  - Chỉ tập trung hỗ trợ kiểm thử trên Android.
  - Không hỗ trợ kiểm thử đa nền tảng.

### *j. AcceLQ*

- Link website: <https://www.accelq.com>

- Mô tả: là một công cụ tự động hóa kiểm thử phần mềm được phát triển để tăng cường quá trình kiểm thử phần mềm. Nó cung cấp các tính năng quản lý kiểm thử, tự động hóa kiểm thử, và theo dõi quá trình kiểm thử, giúp các nhà phát triển phần mềm và nhà kiểm thử có thể tối ưu hóa công việc của mình. AccelQ cung cấp môi trường làm việc đồng nhất cho cả đội ngũ phát triển và kiểm thử, giúp họ cùng làm việc trên cùng một nền tảng và hiểu rõ hơn về quá trình phát triển phần mềm. Điều này có thể giúp giảm thời gian và chi phí của quá trình kiểm thử, đồng thời tăng tính ổn định và chất lượng của sản phẩm phần mềm
- Danh sách chức năng:
  - **Tự động hóa kiểm thử:** cung cấp khả năng tự động hóa kiểm thử cho các ứng dụng web, di động và dịch vụ web.
  - **Tạo và quản lý kịch bản kiểm thử:** Người dùng có thể tạo, quản lý, và thực thi các kịch bản kiểm thử dễ dàng thông qua giao diện trực quan.
  - **Quản lý quy trình kiểm thử:** giúp quản lý quy trình kiểm thử từ đầu đến cuối, từ việc lập kế hoạch kiểm thử đến theo dõi kết quả kiểm thử.
  - **Kết hợp DevOps:** Nó cung cấp tích hợp với các công cụ DevOps phổ biến, giúp tạo môi trường liên tục và tích hợp kiểm thử vào quy trình phát triển.
  - **Xây dựng và quản lý dữ liệu kiểm thử:** cho phép tạo và quản lý dữ liệu kiểm thử để thực hiện kiểm thử với nhiều trường hợp khác nhau.
  - **Báo cáo và theo dõi tiến độ kiểm thử:** Nó cung cấp báo cáo chi tiết về kết quả kiểm thử và theo dõi tiến độ kiểm thử theo thời gian.
  - **Kiểm thử liên tục và tự động:** hỗ trợ kiểm thử liên tục và tự động, giúp giảm thời gian kiểm thử và tăng tính ổn định của sản phẩm.
  - **Kiểm thử đa kênh và đa nền tảng:** Nó hỗ trợ kiểm thử trên nhiều kênh và nền tảng khác nhau, bao gồm ứng dụng web, di động và dịch vụ web.
- Điểm mạnh:
  - Tích hợp dễ dàng: AccelQ cho phép tích hợp dễ dàng với các công cụ DevOps và các công cụ kiểm thử phổ biến khác, giúp tạo ra môi trường liên tục và tích hợp kiểm thử mạnh mẽ.

- Giao diện trực quan và dễ sử dụng: Giao diện người dùng của AccelQ được thiết kế đơn giản và dễ sử dụng, giúp người dùng tạo và quản lý kịch bản kiểm thử một cách dễ dàng.
  - Hỗ trợ đa nền tảng và đa kênh: AccelQ hỗ trợ kiểm thử trên nhiều nền tảng và kênh khác nhau, bao gồm ứng dụng web, di động và dịch vụ web.
  - Tự động hóa toàn diện: Nó cung cấp khả năng tự động hóa toàn diện quá trình kiểm thử, giúp giảm thời gian và công sức cần thiết cho quá trình kiểm thử.
- Điểm yếu:
- Giá cả cao: Một số người dùng có thể cảm thấy giá cả của AccelQ khá cao, đặc biệt là đối với các người dùng cá nhân
  - Yêu cầu kỹ năng kỹ thuật cao: Đôi khi, việc sử dụng AccelQ có thể đòi hỏi người dùng có kiến thức kỹ thuật chuyên sâu để tận dụng tối đa các tính năng và khả năng của nó.
  - Yêu cầu thời gian đào tạo: Để sử dụng AccelQ hiệu quả, người dùng có thể cần một thời gian đào tạo đủ để hiểu rõ về cách sử dụng nó và tối ưu hóa quá trình kiểm thử.

## 6. Bảng so sánh các công cụ kiểm thử.

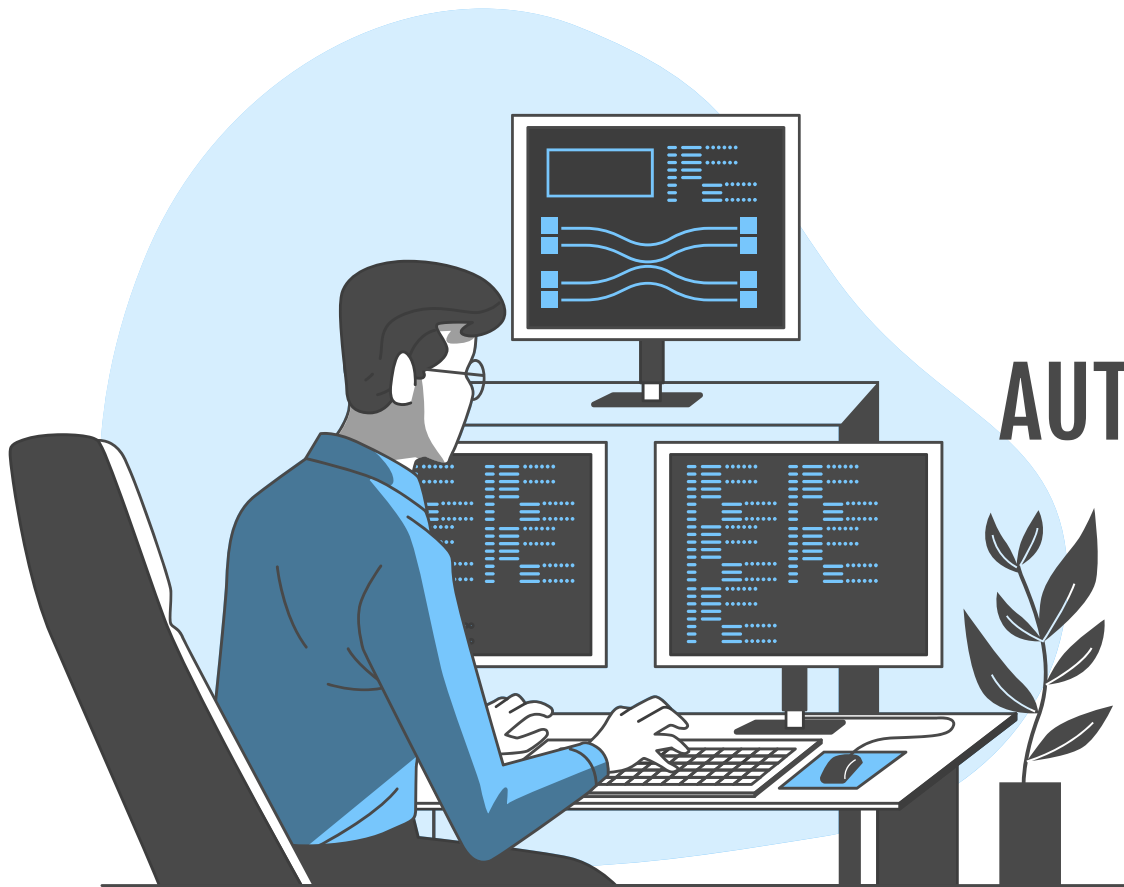
	Katalon	Appnium	TestComplete	Ranorex Studio	Nightwatch.js
Giá cả	Gồm 2 version: Free và Premium	Miễn phí	Đắt đỏ - 3655 đô la	Có trả phí	Miễn phí
Hệ điều hành công cụ hoạt động	Windows , macOS , Linux	Windows, macOS, Linux	Chỉ Windows	Chỉ Windows	Windows , macOS , Linux

Cách tiếp cận (GUI hay code)	GUI và Code	GUI và Code	GUI và Code	GUI và Code	Code
Ngôn ngữ kiểm thử	Groovy và Java	Java, Ruby, Python, PHP, JavaScript, C#	JavaScript, Python, VBScript, JScript, Delphi, C++, C#.	C#, VB.NET	JavaScript, TypeScript
Hỗ trợ Android	Có	Có	Có	Có	Có
Hỗ trợ iOS	Có	Có	Có	Có	Có
Có ghi và phát lại kiểm thử hay không?	Có	Có	Có	Có	Không
Cộng đồng hỗ trợ	Cộng đồng hỗ trợ còn hạn chế	Cộng đồng lớn mạnh	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn

	Espresso	Quantum	Kobiton	AcceLQ	Robotium
Giá cả	Miễn phí	Có trả phí	Trải nghiệm, Có phí	Trải nghiệm, Có phí	Miễn phí



Hệ điều hành công cụ hoạt động	Android	Windows, macOS, Linux	Windows, macOS	Windows, macOS, Linux	Windows, macOS, Linux
Cách tiếp cận (GUI hay code)	Code	Code	GUI, Code	GUI	Code
Ngôn ngữ kiểm thử	Java , Kotlin	Java và JavaScript	C#, Java, Ruby, NodeJS, PHP, or Python	"natural language programming" and application abstraction	Java
Hỗ trợ Android	Có	Có	Có	Có	Có
Hỗ trợ iOS	Không	Có	Có	Có	Không
Có ghi và phát lại kiểm thử hay không?	Không	Không	Có	Có	Không
Cộng đồng hỗ trợ	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn



# AUTOMATION TESTING FOR MOBILE

Nhóm F2 – Oh Bug!

# Table of Contents

01

## Define

Khái niệm automation testing  
và mobile automation testing

02

## Process

Quy trình của mobile  
automation testing

03

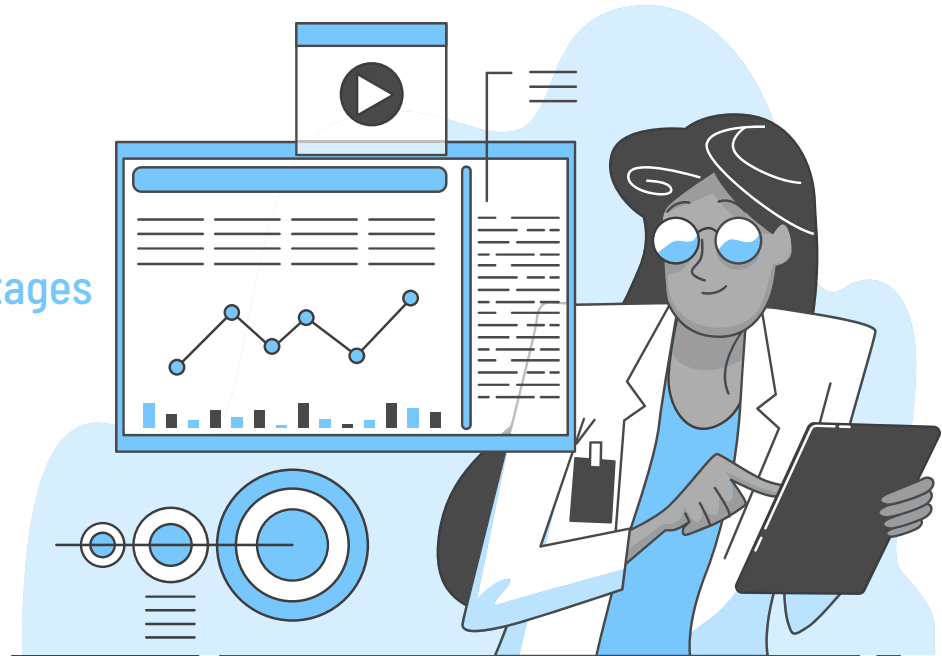
## Advantages and Disadvantages

Lợi ích và nhược điểm của  
mobile automatio testing

04

## Cases

Những trường hợp nào nên  
dùng Automation Testing cho  
ứng dụng Mobile



# Table of Contents

05

## Tools

Một số công cụ kiểm thử tự động  
...

06

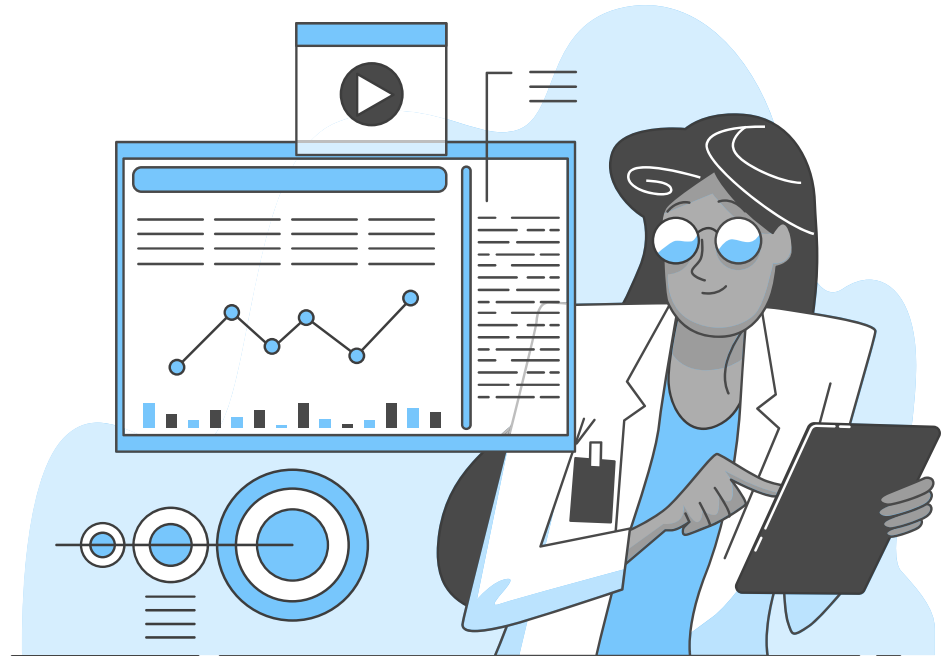
## Compare

Bảng so sánh giữa các công cụ  
...

07

## Demo

Demo các kịch bản kiểm thử với các công cụ.  
...



# 01

## Define

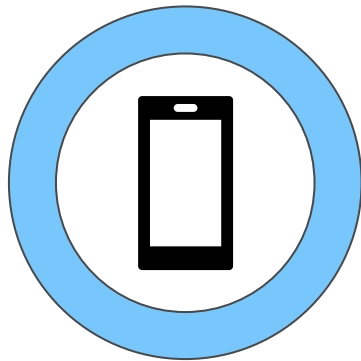
Khái niệm về automation testing và  
mobile automation testing



# Automation Testing

Kiểm thử tự động là việc thực hiện kiểm thử bằng máy mà người kiểm thử sẽ khởi động hệ thống, nhập dữ liệu đầu vào, kiểm tra và so sánh với dữ liệu đầu ra và ghi lại kết quả.

...



# Mobile Automation Testing

Là quá trình kiểm tra chức năng, hiệu suất, khả năng sử dụng và tính nhất quán trên các thiết bị di động như điện thoại di động, máy tính bảng, đảm bảo rằng ứng dụng hoạt động đúng và tương thích trên các thiết bị khác, nền tảng, hệ điều hành khác nhau và trong điều kiện sử dụng khác nhau.

...




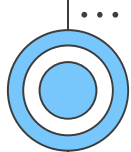


# 02

## Process

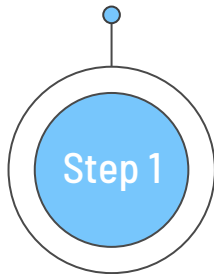
Quy trình thực hiện Automation  
Testing cho ứng dụng di động





# Process

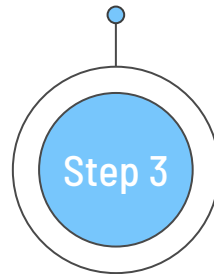
Xác định mục tiêu



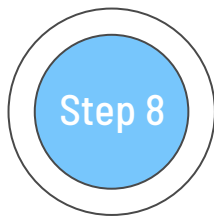
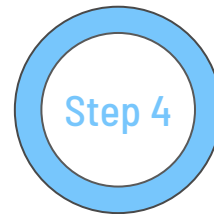
Lựa chọn công cụ



Xây dựng kịch bản



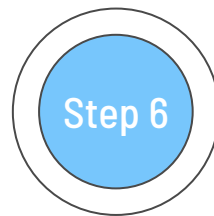
Tự động hóa kiểm thử



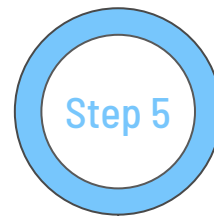
Tối ưu hóa



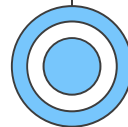
Báo cáo



Phân tích kết quả



Chạy kiểm thử



# 03

## Advantages and Disadvantages

Ưu điểm và nhược điểm mobile  
automation testing

# Advantages



## Tốc độ

Nhất là khi có nhiều trường hợp kiểm thử.



## Tính lặp lại

Kiểm thử tự động có thể chạy nhiều test-case với cùng một quy trình



## Độ tin cậy

Kiểm thử tự động ít xảy ra lỗi của con người hơn



## Tiết kiệm thời gian

Không yêu cầu nỗ lực thủ công cho các tác vụ lặp đi lặp lại



## Tích hợp liên tục

Tích hợp vào quy trình tích hợp liên tục, cho phép phát hiện sớm các lỗi

# Disadvantages



## Chi phí thiết lập ban đầu

Duy trì, bao gồm đầu tư vào công cụ, tập lệnh và cơ sở hạ tầng.



## Độ phức tạp

Có thể phức tạp và khó hiểu đối với các bên liên quan phi kỹ thuật.



## Chi phí bảo trì

Khi công cụ có sự thay đổi, dẫn đến chi phí bảo trì bổ sung



## Kết quả "pass" giả

Bỏ sót một số lỗi, đặc biệt là những lỗi liên quan đến UX.

# 04

## Cases

Những trường hợp nên sử dụng  
Automation Testing cho ứng dụng  
di động



# Cases



## Kịch bản lặp

Các kịch bản kiểm thử mà cần được thực hiện nhiều lần, automation testing có thể giúp tiết kiệm thời gian và tăng tính chính xác



## Tích hợp

Kiểm tra sự tương tác giữa ứng dụng di động và các thành phần hệ thống bên ngoài



## Đa nền tảng, thiết bị

Ứng dụng cần hoạt động trên nhiều loại thiết bị và nền tảng khác nhau







# Cases



## Hiệu suất, độ ổn định

Automation testing có thể tạo ra tải tự động và đo lường hiệu suất theo cách thức được thiết lập trước đó



## Đa ngôn ngữ

Đảm bảo rằng tất cả các phiên bản ngôn ngữ của ứng dụng hoạt động đúng như mong đợi trên mọi thiết bị và hệ điều hành



## Tương thích

Kiểm tra tính tương thích của ứng dụng trên nhiều loại thiết bị di động và phiên bản hệ điều hành khác nhau



# 05 Tools

Một số công cụ kiểm thử tự động

# Katalon Studio



- <https://katalon.com/>
- Katalon Studio giúp kiểm tra tự động các ứng dụng và dịch vụ , cung cấp một môi trường tích hợp để tạo, quản lý và chạy các kịch bản kiểm tra tự động một cách dễ dàng mà không cần biết nhiều về lập trình.

# Katalon Studio



# Katalon

- Giao diện dựa trên GUI: giao diện đồ họa dễ sử dụng để tạo và quản lý các kịch bản kiểm tra
- Hỗ trợ nhiều nền tảng: iOS và Android, và dịch vụ web.
- Báo cáo và lưu trữ kết quả
- Hỗ trợ Kiểm thử đám mây
- Ghi video và phát lại: tạo kịch bản kiểm thử một cách thuận tiện
- Khả năng tích hợp: Git, Jenkins, qTest và Jira

# Katalon Studio



# Katalon

- Cài đặt và sử dụng đơn giản
- Dễ tiếp cận đối với những bạn mới bắt đầu học automation
- Hỗ trợ cho người có chuyên môn về lập trình
- Tạo bài kiểm thử hiệu quả thông qua các template sẵn có
- Kiểm thử đa trình duyệt dễ dàng
- Báo cáo kiểm thử tự động , trực quan

# Katalon Studio



# Katalon

- Cộng đồng support vẫn còn hạn chế
- Chỉ hỗ trợ Groovy và Java
- Performance chậm khi project lớn
- Giới hạn chức năng đối với free version



# APPIUM

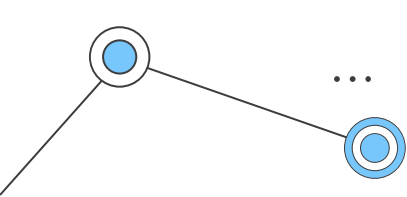
- <https://appium.io/docs/en/2.1/>
- Appium là một công cụ mã nguồn mở được sử dụng để kiểm thử tự động trên các ứng dụng di động trên các nền tảng iOS và Android.
- Appium được xây dựng trên cơ sở của WebDriver. Với kiến trúc đa nền tảng, Appium cho phép viết các kịch bản kiểm thử một lần và chạy chúng trên cả iOS và Android mà không cần sửa đổi mã nguồn.



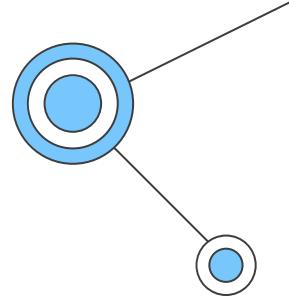
# APPIUM

- Mã nguồn mở và miễn phí
- Hỗ trợ đa nền tảng: iOS và Android.
- Appium hỗ trợ nhiều ngôn ngữ lập trình như Java, Python, Ruby, JavaScript và C#.
- Hỗ trợ các ứng dụng native, hybrid và web.
- Tìm kiếm các thành phần giao diện đa nền tảng.

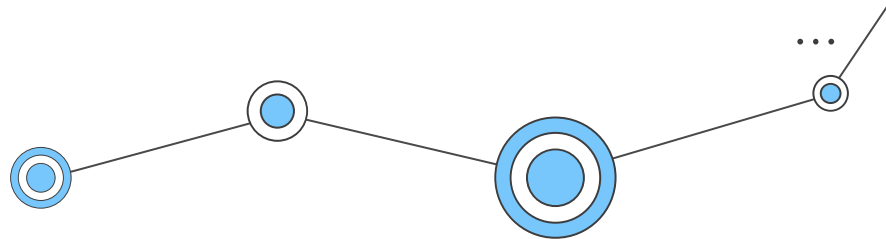




# APPIUM



- Hỗ trợ kiểm thử trên thiết bị thật và giả lập (Emulators và Simulators).
- Tích hợp nhiều Framework kiểm thử: Appium tích hợp các Framework phổ biến như TestNG, JUnit và XCTest.
- Hỗ trợ Kiểm thử trên đám mây (Cloud testing).



# Nightwatch.js



**Nightwatch.js**

- <https://nightwatchjs.org/>
- Nightwatch.js là một framework linh hoạt cho kiểm thử tự động mà mở rộng khả năng của mình để bao gồm kiểm thử cho ứng dụng di động, cả trên nền tảng Android và iOS, thông qua việc tích hợp với Appium.

# Nightwatch.js



**Nightwatch.js**

- Tích hợp với Appium: cho phép kiểm thử tự động trên cả Android và iOS.
- Khung kiểm thử thống nhất: Đảm bảo sự nhất quán giữa các kịch bản kiểm thử trên web và di động.
- Kiểm thử chức năng Từ Đầu Đến Cuối
- Kiểm thử chéo nền tảng
- Ngôn ngữ JavaScript: dễ tiếp cận và sự quen thuộc cho nhà phát triển
- Thực hiện kiểm thử đồng thời

# Nightwatch.js



**Nightwatch.js**

- Hiệu suất chéo nền tảng
- Khung kiểm thử thống nhất
- Tích hợp lệnh toàn diện: Bộ lệnh tích hợp sẵn giúp đơn giản hóa việc tạo kịch bản kiểm thử
- Dễ sử dụng
- Ưu tiên JavaScript
- Thực hiện kiểm thử đồng thời

# Nightwatch.js



**Nightwatch.js**

- Hiệu năng: Giải quyết vấn đề về hiệu năng với các ứng dụng lớn và phức tạp.
- Quản lý trạng thái
- Hạn chế trong xử lý ảnh: liên quan đến xác nhận và so sánh hình ảnh.
- Cộng đồng và tài nguyên hạn chế
- Khả năng mở rộng và tích hợp
- Chưa có sự hỗ trợ mạnh mẽ cho kiểm thử đa ngôn ngữ.

# Espresso



- <https://developer.android.com/training/testing/espresso>
- Espresso là một framework được cung cấp và phát triển bởi google nhằm phục vụ cho việc kiểm thử giao diện trên nền tảng Android.

# Espresso



- Khả năng đồng bộ hóa tự động
- Quản lý hành vi của Android Intents
- Tự động đồng bộ hóa với giao diện người dùng
- Tính linh hoạt và mở rộng cao: API đơn giản, nhẹ, dễ học và dễ bảo trì, với khả năng linh hoạt và mở rộng cao.
- Hỗ trợ kiểm thử unit và black-box
- Module Riêng Biệt Cho Kiểm Thử Android WebView và Android Intents
- Hỗ trợ lấy phản hồi nhanh chóng
- Không sử dụng máy chủ

# Espresso



- Đồng bộ tự động: đồng bộ tự động với giao diện người dùng để thực hiện kiểm thử một cách ổn định và nhanh chóng mà không cần kết nối đến máy chủ.
- Dễ sử dụng: Tích hợp với Android Studio, giúp việc gỡ lỗi và phân tích kết quả kiểm thử trở nên thuận tiện.
- Luồng làm việc đơn giản và phản hồi nhanh chóng
- API ngắn gọn



# Espresso



- Hạn chế về ngôn ngữ lập trình: chỉ bao gồm Java và Kotlin
- Không hỗ trợ chéo nền tảng
- Tập trung chủ yếu vào kiểm thử giao diện người dung: hạn chế đối với dự án lớn hoặc phức tạp đòi hỏi sự phối hợp giữa nhiều đội và môi trường.
- Hơn nữa, việc tự động hóa với Espresso yêu cầu bạn phải có quyền truy cập vào mã nguồn của ứng dụng, có thể làm cho quá trình tự động hóa kiểm thử trở nên phức tạp.

# TestComplete



SMARTBEAR

TestComplete

- <https://smartbear.com/product/testcomplete/overview/>
- Với sự hỗ trợ tích hợp cho hơn 500 cách điều khiển và framework của bên thứ ba, TestComplete có thể xử lý và xác định các thành phần giao diện người dùng động trong hầu hết các công nghệ hiện có.



SMARTBEAR

TestComplete

# TestComplete

- Ngôn ngữ lập trình được hỗ trợ: JavaScript, Python, VBScript, JScript, Delphi, C++, C#.
- Các phương pháp linh hoạt để thiết kế kiểm thử: chế độ ghi và phát lại, chế độ thủ công và tập lệnh với các từ khóa tích hợp.
- Nhận dạng đối tượng bằng nhận dạng hình ảnh dựa trên thuộc tính và AI.
- Kiểm thử song song, đa trình duyệt và đa thiết bị.
- Tích hợp với các khung thử nghiệm, công cụ CI/CD khác và hệ sinh thái SmartBear.

# TestComplete



SMARTBEAR

TestComplete

## Điểm mạnh

- Tính linh hoạt.
- Kiểm thử đa nền tảng.
- Nhận dạng đối tượng và viết kịch bản.
- Ghi và phát lại.
- Bộ thư viện kiểm thử và tích hợp phong phú.
- Trực quan hóa kiểm tra.

# TestComplete



SMARTBEAR

TestComplete

## Điểm yếu

- Chi phí bản quyền.
- Chỉ hỗ trợ Windows.
- Chi phí bảo trì.
- Ghi và phát lại.
- Tùy chọn kiểm thử đám mây hạn chế..

# Ranorex Studio



- <https://www.ranorex.com/>
- Mô tả: Ranorex Studio là một giải pháp tự động hóa kiểm thử phần mềm mạnh mẽ dành cho kiểm thử ứng dụng máy tính, ứng dụng di động và trang web.
- Cung cấp một môi trường đồ họa để tạo và quản lý các kịch bản kiểm thử tự động và hỗ trợ việc kiểm tra trên nhiều nền tảng.

# Ranorex Studio



## Chức năng

- Ghi và phát lại kịch bản kiểm thử tự động
- Hỗ trợ kiểm thử đa nền tảng: Windows, web, Android, iOS và nhiều hệ điều hành khác.
- Quản lý kiểm thử
- Hỗ trợ cho ngôn ngữ lập trình: tùy chỉnh kịch bản kiểm thử bằng cách sử dụng C# hoặc VB.NET.
- Hỗ trợ cho kiểm thử liên tục

# Ranorex Studio



## Điểm mạnh

- Kiểm thử tự động đa nền tảng.
- Tạo test-case mà không cần code: Ranorex Recorder với giao diện kéo và thả cho phép thực hiện các test-case không cần code.
- Sử dụng dễ dàng: Ranorex được coi là một trong những công cụ tự động hóa thử nghiệm dễ tiếp cận nhất trên thị trường.
- Theo dõi image.





# Ranorex<sup>®</sup>

An Idera, Inc. Company

## Ranorex Studio

### Điểm yếu

- Bản quyền trả phí.
- Chỉ có một số ngôn ngữ được hỗ trợ, không giống như Selenium cho phép mã hóa bằng mười ngôn ngữ lập trình phổ biến.
- MacOS không thể cài Ranorex Studio.
- Cộng đồng sử dụng không lớn, chính vì vậy nếu có bug sẽ khó trong việc tìm hiểu.
- Các phiên bản của Ranorex Studio không mang tính ổn định, thường xuyên được cập nhật.



# ACCELQ

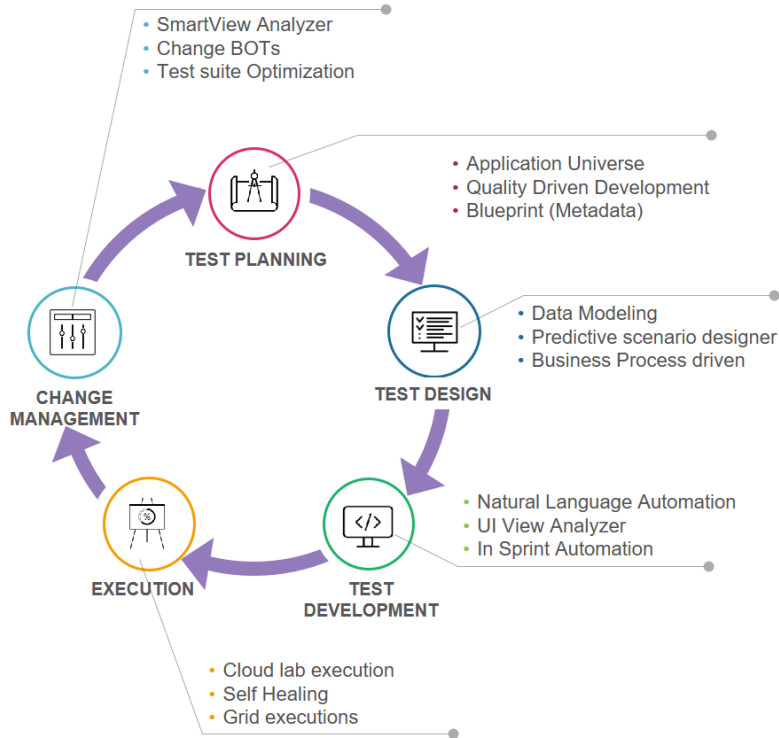


- <https://accelq.com/>
- Là một công cụ tự động hóa kiểm thử phần mềm được phát triển để tăng cường quá trình kiểm thử phần mềm. Nó cung cấp các tính năng quản lý kiểm thử, tự động hóa kiểm thử, và theo dõi quá trình kiểm thử.

## ACCELQ



# ACCELQ



- Tự động hóa kiểm thử
- Tạo và quản lý kịch bản kiểm thử
- Quản lý quy trình kiểm thử
- Kết hợp DevOps
- Xây dựng và quản lý dữ liệu kiểm thử
- Báo cáo và theo dõi tiến độ kiểm thử theo dõi tiến độ kiểm thử theo thời gian
- Kiểm thử liên tục và tự động
- Kiểm thử đa kênh và đa nền tảng



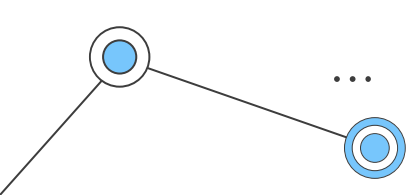
# QUANTUM

- <https://www.perfecto.io/integrations/quantum>
- Quantum là một framework kiểm thử tự động mã nguồn mở được phát triển bởi Perfecto. Nó cung cấp khả năng kiểm thử tự động trên cả ứng dụng di động và trình duyệt web. Quantum hỗ trợ kiểm thử trên các nền tảng di động như iOS và Android.

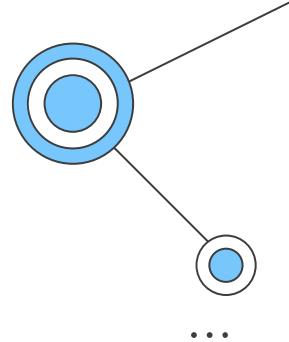


# QUANTUM

- Mã nguồn mở, đơn giản và dễ sử dụng.
- Quantum hỗ trợ kiểm thử trên cả hai nền tảng di động chính là iOS và Android.
- Xử lý sự kiện trên giao diện người dùng ở thiết bị di động
- Chạy kiểm thử đa thiết bị.
- Tạo báo cáo kết quả kiểm thử
- Tích hợp với các công cụ kiểm thử liên tục (CI/CD)

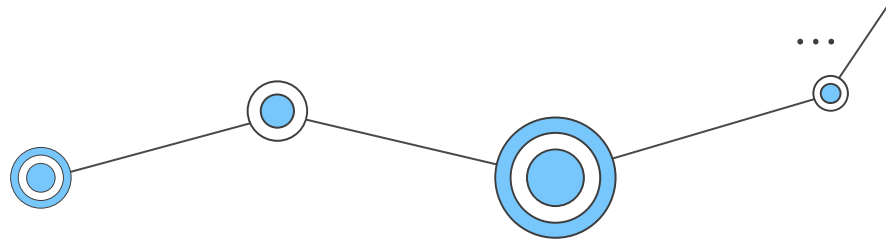


# KOBITON



Kobiton

- <https://kobiton.com/>
- Kobiton là một nền tảng kiểm thử và triển khai ứng dụng di động dựa trên cloud. Nó cung cấp các công cụ và dịch vụ để kiểm thử ứng dụng di động trên nhiều thiết bị và nền tảng khác nhau.





# Kobiton

## KOBITON

- Kobiton cho phép kiểm thử trên nhiều hệ điều hành, bao gồm iOS và Android.
- Tích hợp các cộng cụ Appium và Selenium.
- Hỗ trợ quản lý vòng đời ứng dụng.
- Kobiton giúp quản lý thiết bị từ xa thông qua Cloud.
- Cung cấp các báo cáo và phân tích chi tiết về quá trình kiểm thử.



# Robotium

## ROBOTIUM

- <https://github.com/RobotiumTech/robotium>
- Robotium là một framework kiểm thử mã nguồn mở được tạo ra để giúp việc viết các trường hợp kiểm thử giao diện người dùng (UI) tự động mạnh mẽ và chắc chắn cho ứng dụng di động Android.





# Robotium

## ROBOTIUM

- Là công cụ mã nguồn mở và miễn phí.
- Dễ sử dụng và giúp tiết kiệm thời gian để viết và thực thi.
- Tập trung việc hỗ trợ kiểm thử tự động trên Android.
- Hỗ trợ kiểm thử các ứng dụng native và hybrid.
- Tích hợp các công cụ: Maven, Gradle và Ant.



# 06

## Compare

Bảng so sánh giữa các công cụ



	Katalon	Appnium	TestComplete	Ranorex Studio	Nightwatch.js
<b>Giá cả</b>	Gồm 2 version: Free và Premium	Miễn phí	Đắt đỏ - 3655 đô la	Có trả phí	Miễn phí
<b>Hệ điều hành công cụ hoạt động</b>	Windows, macOS, Linux	Windows, macOS, Linux	Chỉ Windows	Chỉ Windows	Windows , macOS , Linux
<b>Cách tiếp cận (GUI hay code)</b>	GUI và Code	GUI và Code	GUI và Code	GUI và Code	Code
<b>Ngôn ngữ kiểm thử</b>	Groovy và Java	Java, Ruby, Python, PHP, JavaScript, C#	JavaScript, Python, VBScript, JScript, Delphi, C++, C#.	C#, VB.NET	JavaScript, TypeScript
<b>Hỗ trợ Android</b>	Có	Có	Có	Có	Có
<b>Hỗ trợ iOS</b>	Có	Có	Có	Có	Có
<b>Có ghi và phát lại kiểm thử hay không?</b>	Có	Có	Có	Có	Không
<b>Cộng đồng hỗ trợ</b>	Cộng đồng hỗ trợ còn hạn chế	Cộng đồng lớn mạnh	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn

	Espresso	Quantum	Kobiton	AcceLQ	Robotium
<b>Giá cả</b>	Miễn phí	Có trả phí	Trải nghiệm, Có phí	Trải nghiệm, Có phí	Miễn phí
<b>Hệ điều hành công cụ hoạt động</b>	Android	Windows, macOS, Linux	Windows, macOS	Windows, macOS, Linux	Windows, macOS, Linux
<b>Cách tiếp cận (GUI hay code)</b>	Code	Code	GUI, Code	GUI	Code
<b>Ngôn ngữ kiểm thử</b>	Java , Kotlin	Java và JavaScript	C#, Java, Ruby, NodeJS, PHP, or Python	"natural language programming" and application abstraction	Java
<b>Hỗ trợ Android</b>	Có	Có	Có	Có	Có
<b>Hỗ trợ iOS</b>	Không	Có	Có	Có	Không
<b>Có ghi và phát lại kiểm thử hay không?</b>	Không	Không	Có	Có	Không
<b>Cộng đồng hỗ trợ</b>	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn	Cộng đồng hỗ trợ không lớn

# Thanks!

Do you have any questions?

