TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
ĐẠI HỌC QUỐC GIA HÀ NỘI

Báo cáo tìm hiểu công cụ kiểm thử Appium

Nguyễn Đức Phát - 22028298
Nguyễn Sỹ Tân - 22028160
Dương Anh Tú - 22028021

Ngày 21 tháng 4 năm 2025

Mục lục

Danh sách hình vẽ

Danh sách bảng

1 Giới thiệu

1.1 Appium là gì

Appium là một dự án mã nguồn mở và một hệ sinh thái phần mềm liên quan, được thiết kế để hỗ trợ tự động hóa giao diện người dùng (UI) trên nhiều nền tảng ứng dụng, bao gồm di động (iOS, Android, Tizen), trình duyệt (Chrome, Firefox, Safari), máy tính để bàn (macOS, Windows), TV (Roku, tvOS, Android TV, Samsung), và nhiều nền tảng khác [1].

1.2 Lịch sử phát triển của Appium

Appium bắt đầu từ một ý tưởng của Dan Cuellar, Test Manager tại Zoosk vào năm 2011, khi đối mặt với vấn đề quá tải trong việc kiểm thử ứng dụng iOS. Sau nhiều thử nghiệm với các công cụ hiện có, Dan phát triển iOSAuto, một công cụ tự động hóa sử dụng cú pháp giống Selenium. Sau khi Dan trình bày tại Selenium Conference 2012, dự án này thu hút sự chú ý từ cộng đồng và Jason Huggins, đồng sáng lập Selenium.

Vào tháng 8 năm 2012, Dan quyết định phát hành mã nguồn của iOSAuto dưới giấy phép mã nguồn mở trên GitHub, và sau đó đổi tên thành Appium. Appium nhanh chóng được sự hỗ trợ từ Sauce Labs và chuyển sang nền tảng Node.js vào đầu năm 2013 để phát triển và mở rộng cộng đồng. Từ đó, Appium trở thành một công cụ tự động hóa đa nền tảng với sự hỗ trợ cho cả iOS và Android.

Đến tháng 5 năm 2014, Appium 1.0 chính thức ra mắt, đánh dấu cột mốc quan trọng trong sự phát triển của dự án. Appium đã trở thành framework tự động hóa di động mã nguồn mở phổ biến nhất, được sử dụng rộng rãi trong ngành và tiếp tục phát triển nhờ sự đóng góp của cộng đồng và sự hỗ trợ từ Sauce Labs.

Appium 2.0 đã được phát hành vào năm 2022, với trọng tâm mới là xem Appium như một hệ sinh thái thay vì chỉ là một dự án đơn lẻ. Các trình điều khiển (drivers) và plugin có thể được phát triển và chia sẻ bởi bất kỳ ai, mở ra một thế giới đầy tiềm năng cho việc phát triển liên quan đến tự động hóa trên nhiều nền tảng vượt ngoài iOS và Android.

Báo cáo này sẽ tập trung vào việc phân tích và kiểm thử với phiên bản Appium 2.x (bao gồm các phiên bản nâng cấp của Appium 2.0).

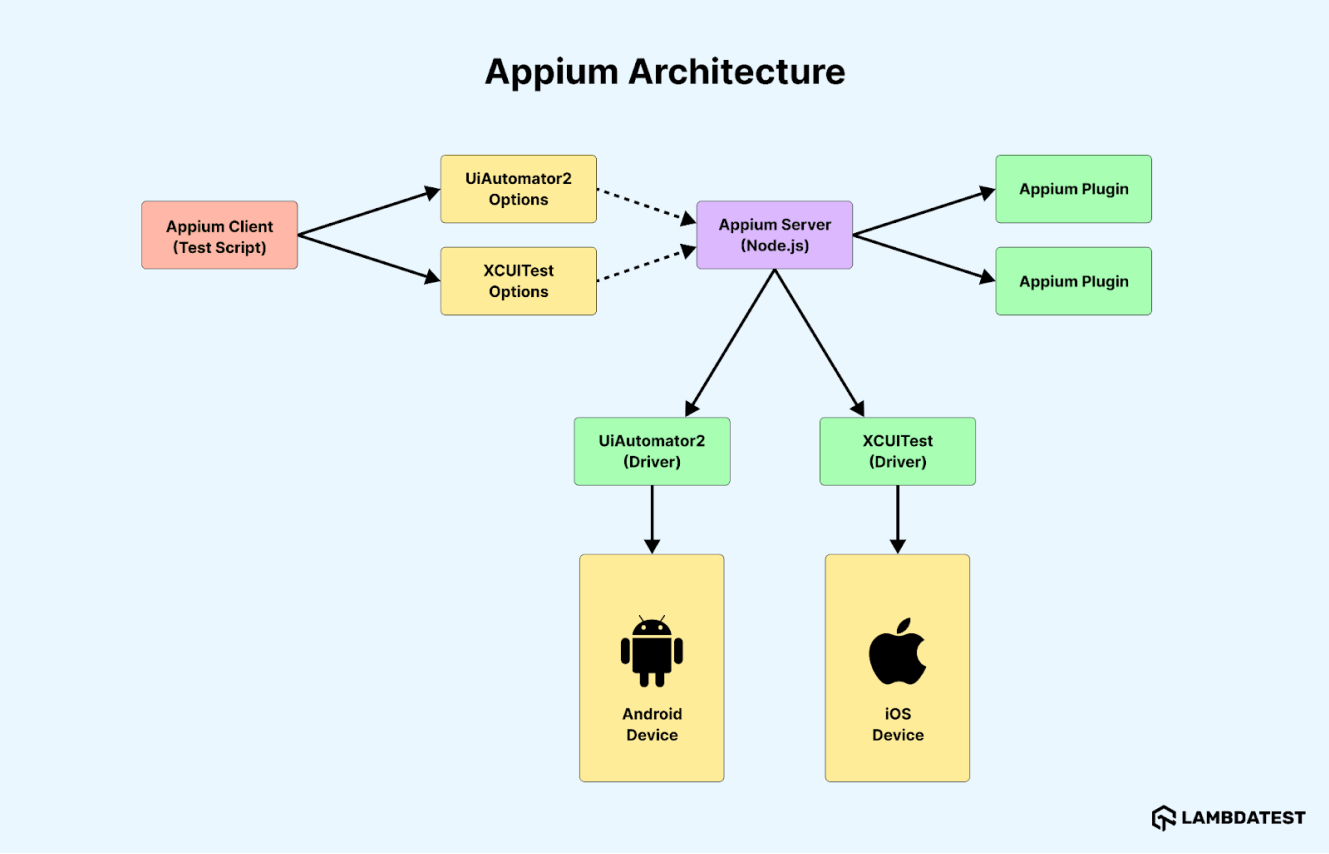
1.3 Tại sao nên chọn Appium

- Appium giúp không cần phải biên dịch lại ứng dụng hoặc chỉnh sửa nó theo bất kỳ cách nào, nhờ vào việc sử dụng các API tự động hóa tiêu chuẩn trên tất cả các nền tảng.
- Nhà phát triển có thể viết các bài kiểm thử bằng công cụ phát triển yêu thích của mình, sử dụng bất kỳ ngôn ngữ nào tương thích với WebDriver như Java, Python, Ruby và C#. Ngoài ra, còn có các bản triển khai client bên thứ ba cho các ngôn ngữ khác.
- Appium hỗ trợ sử dụng bất kỳ framework kiểm thử nào.
- Một số trình điều khiển như `xcuitest` và `uiautomator2` có hỗ trợ sẵn cho ứng dụng web di động và ứng dụng lai (hybrid). Trong cùng một script, bạn có thể chuyển đổi mượt mà giữa tự động hóa ứng dụng gốc và tự động hóa trong webview, tất cả đều sử dụng mô hình WebDriver — mô hình đã trở thành tiêu chuẩn cho tự động hóa web.

- Nhà phát triển có thể chạy các bài kiểm thử tự động của mình cả ở local lẫn trên cloud. Có nhiều nhà cung cấp dịch vụ đám mây hỗ trợ các driver của Appium (chủ yếu nhắm đến tự động hóa di động cho iOS và Android).
- `Appium Inspector` có thể được sử dụng để kiểm tra trực quan mã nguồn giao diện của ứng dụng trên các nền tảng khác nhau, giúp việc phát triển bài kiểm thử trở nên dễ dàng hơn.

2 Kiến trúc và cách thức hoạt động của công cụ

2.1 Kiến trúc của Appium



Hình 1: Kiến trúc Appium 2.x

Vì Appium dựa trên thông số kỹ thuật WebDriver của W3C , nên nó triển khai kiến trúc client-server. Cấu trúc của Appium bao gồm các thành phần chính: Appium Server, Appium Client, Appium Driver, Appium Plugin và thiết bị đích [2]. Trong đó Appium Driver và Plugin mới chỉ được xuất hiện ở các phiên bản v2.x.

2.1.1 Appium Server

Appium Server là một API server viết bằng Node.js. Server này cần được chạy trên thiết bị đích nơi các thiết bị di động được kết nối. Nó hoạt động như là điểm liên lạc cho kịch bản tự động hóa,

nơi các ràng buộc của Appium Client sẽ truyền các chi tiết về thiết bị đích và ứng dụng thông qua các tùy chọn khả năng (capability). Appium Server sẽ phân tích các tùy chọn mà Appium Client cung cấp để xác định trên thiết bị di động nào có sẵn để bắt đầu phiên làm việc.

2.1.2 Appium Client

Appium client chứa mã tự động hóa theo kịch bản được viết bằng các ngôn ngữ lập trình phổ biến như Java, Python, Ruby, Perl, v.v. Các tập lệnh tự động hóa chứa thông tin chi tiết về cấu hình của thiết bị và ứng dụng. Các thông tin chi tiết về kịch bản và cấu hình này được sử dụng để chạy các test cases.

Client mà được sử dụng để viết script kiểm thử sẽ gửi các lệnh đến Appium server qua mạng HTTP. Server sẽ thực thi lệnh và trả lại mã trạng thái cho client trong phản hồi, mà client sẽ sử dụng để xác định xem lệnh đã được server thực thi thành công hay có lỗi xảy ra.

Các lệnh mà Appium client gửi đến Appium server thực tế là các API endpoint cho từng lệnh. Vì API không phụ thuộc vào ngôn ngữ, nên không quan trọng đang sử dụng ngôn ngữ lập trình nào. Đó là lý do tại sao Appium hỗ trợ nhiều liên kết client khác nhau.

2.1.3 Appium Driver

Các driver của Appium là cầu nối giữa Appium Client và thiết bị đích. Appium Server cơ bản xử lý việc giao tiếp giữa Appium Client và Appium Driver. Các lệnh Appium mà Appium Client thực thi sẽ được chuyển tiếp đến driver Appium tương ứng, driver này sẽ bắt đầu phiên làm việc với máy chủ Appium.

Các driver của Appium thực hiện các phương thức API theo các thông số kỹ thuật của WebDriver. Mỗi driver sẽ sử dụng các công nghệ gốc của nền tảng tương ứng khi triển khai driver. Ví dụ, driver `XCUITest` sẽ sử dụng WebDriverAgent, được phát triển bằng ngôn ngữ lập trình Swift và các lệnh gốc của Xcode, còn `UiAutomator2` sẽ sử dụng các phương thức gốc của Android trong các driver tương ứng.

Với Appium v2.x, Appium đã hoàn toàn tách rời các driver khỏi Appium Server như trong các phiên bản trước đó. Giờ đây, chỉ cần cài đặt những driver cần thiết.

2.1.4 Appium Plugin

Appium cung cấp một phương pháp mở rộng và tùy chọn mới để mở rộng và sửa đổi hành vi của Appium trong suốt phiên làm việc, từ đó thêm các tính năng mới và tùy chỉnh cho các tester mà không cần phải sửa đổi trực tiếp Appium Server. Tính năng này của Appium được gọi là Appium plugins.

Với Appium Plugin, chúng ta có thể tạo ra triển khai của riêng mình cho plugin, giúp giải quyết một vấn đề cụ thể hoặc thêm một tính năng mới cho Appium. Nếu không muốn tạo plugin Appium của riêng mình, chúng ta có thể sử dụng các plugin đã được tạo sẵn bởi các contributor khác hoặc các nhà phát triển của Appium. Một số plugin Appium hữu ích có thể sử dụng là *appium-device-farm* và *appium-dashboard-plugin.*

2.1.5 Thiết bị đích

Đây là các thiết bị mô phỏng, thiết bị giả lập hoặc có thể là thiết bị vật lý được kết nối với Appium Server, đây là nơi mà các test script được thực thi.

2.2 Luồng hoạt động của Appium

Appium sử dụng UIAutomator API đối với Android, còn đối với các thiết bị iOS, Appium sử dụng XCUITest API để tương tác giữa các phần tử giao diện của thiết bị đang được kiểm thử. Cách thức hoạt động nhìn chung gồm các pha như sau, được biểu thị qua Hình 1 :

- Trước tiên, Appium server phải được khởi động trên thiết bị xảy ra hành vi kiểm thử tự động, từ đó mới lắng nghe được yêu cầu từ ứng dụng client thông qua giao thức chuẩn Mobile JSON Wire.
- Chương trình client được viết bởi người dùng được chuyển đổi thành các yêu cầu REST API thông qua các thư viện client. Sau đó, yêu cầu được gửi tới Appium server như các đối tượng JSON thông qua Mobile JSON wire protocol.
- Appium server sử dụng giao thức WebDriver, gửi những yêu cầu này đến thiết bị đang được kiểm thử (thiết bị di động hoặc một trình giả lập).
- Sau khi yêu cầu được gửi, Appium server tương tác với thiết bị Android thông qua Bootstrap.jar. Khi máy chủ tạo một phiên làm việc cho Appium driver, file Bootstrap.jar được đẩy vào thiết bị, cho đến khi thiết bị nhận được yêu cầu, file này được thực thi, nó sẽ khởi động một server lắng nghe trên port mặc định là 4724. Server này lắng nghe yêu cầu đến từ máy chủ Appium, Bootstrap.jar chuyển đổi chúng thành các định dạng Android có thể hiểu được - định dạng UIAutomator nhờ sử dụng API của UIAutomator. Nhờ đó giúp thực hiện các thao tác tự động trên thiết bị Android, bao gồm việc tìm kiếm và tương tác với các phần tử giao diện người dùng.
- Còn các thiết bị iOS tương tác với Appium bằng WebDriverAgent.app để chuyển đổi những đối tượng yêu cầu sang định dạng mà di động có thể hiểu được là XCUITest. Khi Appium tương tác với thiết bị iOS, đầu tiên Appium sẽ kiểm tra sự tồn tại của WebDriverAgent.app, nếu app này không tồn tại, thiết bị sẽ tự động tải WebDriverAgent.app về, kết nối nó với XCUITest.framework và Apple's API để thực thi yêu cầu.
- Kết quả kiểm thử được gửi lại về Appium server thông qua Bootstrap.jar(Android) hoặc WebDriverAgent.app (iOS). Từ server, phản hồi được gửi tiếp về với khách hàng.

3 Cách cài đặt và sử dụng công cụ

3.1 Yêu cầu hệ thống

Appium là một công cụ mã nguồn mở giúp tự động hóa kiểm thử ứng dụng di động trên nhiều nền tảng (Android, iOS). Để sử dụng Appium phiên bản 2.x, hệ thống cần đáp ứng yêu cầu sau:

- **Hệ điều hành**: Windows 10 trở lên, macOS, hoặc các bản phân phối Linux phổ biến.
- **Node.js**: Phiên bản 16 trở lên.
- **JDK**: Java Development Kit từ phiên bản 8 trở lên.
- **Android Studio** (test Android): Bao gồm SDK, AVD, Emulator, drivers.
- **Xcode** (test iOS): Dành cho macOS.
- **Driver Appium**: uiautomator2 cho Android, xcuitest cho iOS.

3.2 Hướng dẫn cài đặt Appium

Appium server có thể được cài đặt thông qua npm:

```
npm install -g appium
npm install -g appium-doctor
```

Sau khi cài đặt có thể xác minh bằng lệnh:

```
appium -v
appium-doctor
```

3.3 Cách viết và chạy một test script với Appium

Để viết một bài kiểm tra Appium bằng JavaScript (Node.js), chúng ta cần chọn một thư viện máy khách tương thích với Appium **app**. Thư viện được bảo trì tốt nhất và là thư viện mà nhóm Appium khuyên dùng là WebdriverIO [3]. Tiếp theo, chỉ cần tạo một thư mục dự án mới ở đâu đó trên máy tính và sau đó khởi tạo một dự án Node.js mới trong đó:

```
npm init
```

Bây giờ, hãy cài đặt webdriveriogói thông qua NPM

```
npm i --save-dev webdriverio
```

Tiếp đến,tạo một tệp mới có tên là test.js với nội dung sau:

```
const {remote} = require('webdriverio');

const capabilities = {
  platformName: 'Android',
  'appium:automationName': 'UiAutomator2',
  'appium:deviceName': 'Android',
  'appium:appPackage': 'com.android.settings',
  'appium:appActivity': '.Settings',
};

const wdOpts = {
  hostname: process.env.APPIUM_HOST || 'localhost',
  port: parseInt(process.env.APPIUM_PORT, 10) || 4723,
  logLevel: 'info',
  capabilities,
};

async function runTest() {
  const driver = await remote(wdOpts);
  try {
    const batteryItem = await driver.$('//*[@text="Battery"]');
    await batteryItem.click();
  } finally {
    await driver.pause(1000);
    await driver.deleteSession();
  }
```

```
}

runTest().catch(console.error);
```

Đoạn mã này thực hiện các bước sau:

- Định nghĩa một tập hợp các "Capabilities"(tham số cấu hình) để gửi đến Appium Server, nhằm thông báo cho Appium biết loại thiết bị và ứng dụng nào cần được tự động hóa.
- Khởi tạo một phiên làm việc (session) với Appium, hướng tới ứng dụng "Cài đặt"(Settings) được tích hợp sẵn trên hệ điều hành Android.
- Tìm kiếm mục danh sách có tên 'Battery' và thực hiện thao tác nhấn vào đó.
- Tạm dừng trong một khoảng thời gian ngắn (1 giây), mục đích chỉ để tạo hiệu ứng trực quan rõ ràng hơn.
- Kết thúc phiên làm việc với Appium.

Trước khi thực thi đoạn mã, cần lưu ý rằng một phiên Appium Server phải đang hoạt động ở một phiên dòng lệnh (terminal) khác. Nếu không, hệ thống sẽ không thể kết nối được với Appium Server và sẽ phát sinh lỗi.

Sau khi bảo đảm Appium Server đang chạy, có thể thực thi đoạn mã bằng lệnh sau:

```
node test.js
```

4 Case study cho Appium

4.1 Các tình huống kiểm thử phổ biến

Trong lĩnh vực kiểm thử phần mềm di động, Appium được sử dụng rộng rãi nhờ khả năng hỗ trợ đa nền tảng, không phụ thuộc ngôn ngữ lập trình của ứng dụng, và tính linh hoạt trong việc tích hợp vào các hệ thống kiểm thử tự động. Dưới đây là một số tình huống kiểm thử phổ biến mà Appium thường được áp dụng:

4.1.1 Kiểm thử chức năng (Functional Testing)

Đây là loại kiểm thử phổ biến nhất, nhằm đảm bảo rằng các chức năng cốt lõi của ứng dụng hoạt động đúng như mong đợi. Appium cho phép mô phỏng các thao tác của người dùng như chạm, kéo thả, nhập văn bản, nhấn nút, chuyển màn hình, và nhiều hành động tương tác khác.

Ví dụ: Kiểm tra quá trình đăng nhập, gửi biểu mẫu, thêm sản phẩm vào giỏ hàng, hoặc thực hiện thanh toán.

4.1.2 Kiểm thử giao diện người dùng (UI Testing)

Appium hỗ trợ xác định các phần tử giao diện thông qua XPath, Accessibility ID hoặc các phương pháp định vị khác, giúp đánh giá tính toàn vẹn và khả năng tương tác của giao diện ứng dụng.

Ví dụ: Kiểm tra xem các nút có hiển thị đúng, vị trí phần tử có đúng thiết kế hay không, hoặc khi xoay màn hình thì bố cục có bị lỗi hay không.

4.1.3 Kiểm thử hồi quy (Regression Testing)

Khi ứng dụng có sự thay đổi về tính năng hoặc giao diện, kiểm thử hồi quy là cần thiết để đảm bảo rằng các chức năng hiện tại không bị ảnh hưởng. Appium giúp tự động hoá quá trình kiểm thử hồi quy, giúp tiết kiệm thời gian và công sức.

Ví dụ: Sau khi thêm tính năng "ưu đãi khuyến mãi", kiểm tra lại toàn bộ quy trình mua hàng để đảm bảo không phát sinh lỗi.

4.1.4 Kiểm thử đa thiết bị (Cross-device Testing)

Ứng dụng di động cần được thử nghiệm trên nhiều thiết bị có kích thước màn hình, độ phân giải và hệ điều hành khác nhau. Appium cho phép cùng một bộ test case có thể chạy được trên nhiều loại thiết bị, giả lập hoặc thiết bị thật.

Ví dụ: Kiểm thử tính năng camera trên các thiết bị Android khác nhau như Samsung, Xiaomi và Pixel.

4.1.5 Kiểm thử cử chỉ người dùng (Gesture Testing)

Appium hỗ trợ kiểm thử các thao tác nâng cao như vuốt, kéo, zoom, double tap, long press,... giúp kiểm tra các hành vi tương tác phức tạp trong ứng dụng.

Ví dụ: Kiểm tra khả năng zoom ảnh bằng hai ngón tay trong một ứng dụng xem ảnh.

4.1.6 Kiểm thử tích hợp với hệ thống CI/CD

Appium có thể dễ dàng tích hợp với các công cụ CI/CD như Jenkins, GitHub Actions hoặc GitLab CI để chạy kiểm thử tự động sau mỗi lần cập nhật mã nguồn. Điều này đảm bảo chất lượng phần mềm được duy trì xuyên suốt vòng đời phát triển.

4.2 Thử nghiệm kiểm thử một ứng dụng di động cụ thể

4.2.1 Giới thiệu ứng dụng

Ứng dụng được kiểm thử là ApiDemos-debug.apk, một ứng dụng mẫu của Appium cung cấp các thành phần UI và tương tác khác nhau để phục vụ kiểm thử tự động. Bài kiểm thử tập trung vào tính năng "Bouncing Balls"nằm trong menu Animation.

4.2.2 Công cụ sử dụng

- Ngôn ngữ lập trình: JavaScript
- Framework: WebdriverIO + Appium
- Thiết bị kiểm thử: Android Emulator
- Tự động hóa: Sử dụng chuỗi hành động performActions để mô phỏng tương tác chạm, kéo và vẽ trên màn hình.
- Ghi nhận kết quả: Ảnh chụp màn hình được lưu lại sau mỗi kịch bản kiểm thử để phục vụ đánh giá kết quả.

4.2.3 Cấu hình

Cấu hình Appium:

```
const capabilities = {
  platformName: "Android",
  "appium:automationName": "UiAutomator2",
  "appium:deviceName": "Android␣Emulator",
  "appium:app": "ApiDemos-debug.apk",
  "appium:appPackage": "io.appium.android.apis",
  "appium:appActivity": ".ApiDemos",
};
```

4.2.4 Các kịch bản kiểm thử

Kịch bản 1: Tap vào 4 góc màn hình

- Mục tiêu: Xác minh ứng dụng phản hồi với các thao tác chạm tại các vị trí khác nhau trên màn hình.
- Thực hiện: Tap vào 4 góc (trên-trái, trên-phải, dưới-trái, dưới-phải).
- Kết quả mong đợi: Quả bóng được tạo tại mỗi vị trí tương ứng.

Kịch bản 2: Kéo theo các hướng

- Mục tiêu: Kiểm tra ứng dụng phản hồi với thao tác kéo liên tục.
- Thực hiện:
  - Kéo đường thẳng ngang
  - Kéo đường thẳng dọc
  - Kéo đường chéo
- Kết quả mong đợi: Quả bóng tạo ra và rơi theo các tương tác kéo tương ứng.

Kịch bản 3: Vẽ hình tròn

- Mục tiêu: Kiểm tra độ mượt của thao tác vẽ đường cong.
- Thực hiện: Vẽ một đường tròn 360° quanh tâm màn hình bằng các điểm tính toán từ công thức hình tròn.
- Kết quả mong đợi: Quả bóng tạo ra dọc theo đường tròn được vẽ.

Kịch bản 4: Vẽ hình chữ Z

- Mục tiêu: Mô phỏng vẽ ký tự hình học phức tạp.
- Thực hiện: Vẽ hình chữ Z bằng ba đoạn thẳng nối tiếp (trái sang phải, chéo, trái sang phải).
- Kết quả mong đợi: Bóng được tạo theo đường vẽ hình chữ Z.

Kịch bản 5: Vẽ hình chữ nhật

- Mục tiêu: Kiểm tra thao tác kéo theo hình học khép kín.
- Thực hiện: Vẽ một chữ nhật theo 4 cạnh.
- Kết quả mong đợi: Bóng được vẽ theo hình vuông.

Kịch bản 6: Tap nhanh liên tiếp nhiều lần

- Mục tiêu: Đánh giá khả năng phản hồi với nhiều thao tác liên tục trong thời gian ngắn.
- Thực hiện: Tap nhanh 10 lần tại các vị trí ngẫu nhiên trên màn hình.
- Kết quả mong đợi: 10 quả bóng xuất hiện tương ứng với số lần tap.

Kịch bản 7: Vẽ hình zigzag

- Mục tiêu: Kiểm tra khả năng nhận diện chuỗi thao tác lặp lại có tính đối xứng.
- Thực hiện: Vẽ một hình zigzag bao gồm 6 điểm tạo thành hình răng cưa.
- Kết quả mong đợi: Các bóng được tạo dọc theo các điểm zigzag.

4.2.5 Ảnh minh hoạ kiểm thử

Ảnh chụp màn hình của từng kịch bản được lưu tự động trong thư mục screenshots/, giúp đối chiếu trực quan với các hành vi mong đợi. Dưới đây là 1 vài hình ảnh về các ca kiểm thử:

(a) Vẽ hình chữ nhật (b) Vẽ hình chữ Z (c) Vẽ đường chéo

Hình 2: Một vài hình ảnh về các ca kiểm thử

5 So sánh với các công cụ khác

Tiêu chí	Appium	Selendroid	Katalon
Nền tảng hỗ trợ	Android, iOS	Android	Android, iOS
Ngôn ngữ lập trình hỗ trợ	Java, Python, Ruby, JS, C#, v.v.	Java	Groovy (Tương tự Java)
Giao diện người dùng	Không có	Không có	Có
Tích hợp CI/CD	Jenkins, GitLab, Bamboo, Azure, CircleCI...	Jenkins	Jenkins, GitLab, Azure, Docker, Katalon TestOps
Thiết bị đích	Thiết bị giả lập, Thiết bị thực	Thiết bị giả lập, Thiết bị thực	Thiết bị giả lập, Thiết bị thực
Hỗ trợ script Selenium	Có (dùng WebDriver protocol)	Có	Có
Kiểu kiến trúc	Client-Server	Nhúng vào ứng dụng	GUI + Framework tích hợp
Mã nguồn mở	Có (Apache 2.0 License)	Có (Apache 2.0 License)	Không (phiên bản Community miễn phí, Pro tính phí)
Khả năng mở rộng	Cao (do hỗ trợ nhiều ngôn ngữ và công cụ CI/CD)	Thấp (do đã ngừng phát triển)	Cao (hỗ trợ CI/CD, test suite mạnh mẽ)
Tình huống kiểm thử phù hợp	Cần kiểm thử ứng dụng Android & iOS	Dự án Android cũ, không cần hỗ trợ iOS	Tester không chuyên lập trình, cần giao diện GUI

Bảng 1: So sánh giữa Appium với một số công cụ kiểm thử

6 Đánh giá về công cụ

6.1 Ưu điểm

- Appium hoàn toàn là một cross platform tự động nguồn mở và miễn phí trong việc sử dụng cho cả các thiết bị iOS, Android hay các ứng dụng Native và ứng dụng Hybrid.
- **Hỗ trợ đa nền tảng**: Appium hỗ trợ kiểm thử ứng dụng trên nhiều nền tảng, từ các ứng dụng web trên trình duyệt di động, iOS, Android- giúp tiết kiệm thời gian và công sức cho việc phát triển kiểm thử. Với một bộ test scripts, chúng có thể chạy được cả trên iOS và Android mà không cần sửa đổi code.
- **Hỗ trợ nhiều ngôn ngữ lập trình**: Appium cho phép sử dụng nhiều ngôn ngữ lập trình phổ biến như Java, Python, JavaScript, Ruby để viết kịch bản kiểm thử, giúp các nhóm kiểm thử dễ dàng sử dụng
- **Hỗ trợ đa dạng các loại ứng dụng**: Appium có khả năng kiểm thử cả ứng dụng di động native và hybrid, thêm vào đó là cả web app- giúp đơn giản hóa quy trình kiểm thử, tăng khả năng sử dụng lại code khi kiểm thử cả native và web app trên cùng một thiết bị

6.2 Nhược điểm

- Hạn chế lớn nhất của Appium đó là người dùng phải trực tiếp viết các ca kiểm thử, vì vậy sử dụng Appium đòi hỏi họ phải có trình độ nhất định đối với ngôn ngữ lập trình được sử dụng trong kịch bản kiểm thử.
- Nhược điểm thứ hai là quá trình cài đặt, thiết lập Appium quá phức tạp, tiềm tàng nhiều khả năng gây lỗi. Để cài đặt Appium, người dùng cần phải cài đặt và cấu hình một loạt các phần mềm và công cụ như Node.js, Android SDK (đối với việc kiểm thử ứng dụng Android), Xcode (đối với việc kiểm thử ứng dụng iOS), Appium Server, và các gói phụ thuộc khác. Quá trình này có thể phức tạp và đòi hỏi kiến thức kỹ thuật và kinh nghiệm cụ thể
- Có thể Appium không hỗ trợ các hệ điều hành và các mẫu thiết bị di động cũ, gây ra khó khăn trong việc kiểm tra các ứng dụng cũ.
- Quá trình thực hiện kiểm thử của Appium có thể chậm hơn so với các framework kiểm thử tự động khác. Hiệu suất tổng thể của quá trình kiểm thử có thể bị ảnh hưởng do thời gian thực hiện kiểm thử bị trì hoãn do các yếu tố như độ phức tạp của việc kết nối với thiết bị di động và chi phí giao tiếp của giao thức WebDriver.

6.3 Tiềm năng phát triển trong tương lai

Appium, với vai trò là một công cụ kiểm thử tự động mã nguồn mở, đang đứng trước nhiều cơ hội phát triển mạnh mẽ trong tương lai. Dưới đây là một số xu hướng và tiềm năng đáng chú ý của công cụ kiểm thử này [4]:

- **Tăng cường tự động hóa với AI và Machine Learning:** Appium đang hướng tới việc tích hợp các công nghệ trí tuệ nhân tạo (AI) và học máy (Machine Learning) để nâng cao khả năng tự động hóa. Điều này sẽ giúp phát hiện lỗi nhanh chóng và chính xác hơn, đồng

thời tối ưu hóa quy trình kiểm thử bằng cách dự đoán và xử lý các vấn đề tiềm ẩn trước khi chúng ảnh hưởng đến người dùng.

- **Mở rộng khả năng kiểm thử đa nền tảng:** Với sự phát triển liên tục của các nền tảng di động, Appium đang được cải tiến để hỗ trợ kiểm thử trên nhiều hệ điều hành và thiết bị khác nhau một cách liền mạch. Điều này giúp giảm thiểu thời gian và công sức trong việc viết và duy trì các kịch bản kiểm thử riêng biệt cho từng nền tảng.
- **Hỗ trợ kiểm thử cho thiết bị IoT và thiết bị thông minh:** Khi Internet of Things (IoT) ngày càng phổ biến, nhu cầu kiểm thử các ứng dụng trên thiết bị thông minh như đồng hồ, tủ lạnh, hay robot hút bụi cũng tăng lên. Appium đang được phát triển để hỗ trợ kiểm thử trên các thiết bị này, đảm bảo tính ổn định và hiệu suất của ứng dụng trong môi trường đa dạng.
- **Cộng đồng phát triển mạnh mẽ và hỗ trợ liên tục:** Một trong những điểm mạnh của Appium là cộng đồng người dùng và nhà phát triển rộng lớn. Sự đóng góp liên tục từ cộng đồng giúp Appium luôn được cập nhật với các xu hướng và công nghệ mới, đồng thời cung cấp tài liệu và hỗ trợ kỹ thuật phong phú cho người dùng.

Tài liệu

[1] A. Authors, *Appium Documentation*, https://appium.io/docs/en/latest/, 2024.

[2] LambdaTest, *Understanding Appium Architecture: Key Components Explained*, https://www.lambdatest.com/blog/appium-architecture/, 2024.

[3] BrowserStack, *Get Started with Appium Testing*, https://www.browserstack.com/guide/appium-tutorial-for-testing, 2024.

[4] T. Q. Lab, *What is the Future of Mobile Application Testing (Appium)?* https://testriq.com/blog/post/what-is-the-future-of-mobile-application-testing-appium, 2024.