

**ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



**BÁO CÁO BÀI TẬP LỚN
KIỂM THỦ VÀ ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM**

**ĐỀ TÀI: CÔNG CỤ KIỂM THỦ TỰ ĐỘNG ỦNG
DỤNG DI ĐỘNG APPiUM**

Giảng viên: Nguyễn Thu Trang

Nhóm 3:

Nguyễn Thanh Huyền - 18020666

Hoàng Quốc Việt - 18020062

Lại Ngọc Tân - 18020049

Giới thiệu

Trong những năm gần đây, điện thoại di động đã trở nên phổ biến và trở thành vật bất ly thân của mỗi người. Số lượng điện thoại đã vượt qua dân số toàn cầu, thu hút người dùng với vô vàn các ứng dụng hữu ích từ giao tiếp liên lạc đến giao dịch thanh toán. Sự phát triển bùng nổ này yêu cầu các nhà phát triển ứng dụng cần bắt kịp tốc độ, tạo ra các sản phẩm ứng dụng hữu ích và đảm bảo chất lượng. Theo dòng thời gian, các ứng dụng ngày càng lớn và phức tạp hơn, nếu kiểm thử một cách thủ công thì gây ra khó khăn, tốn thời gian và mất nhiều chi phí, từ đó việc tự động hóa kiểm thử các ứng dụng càng trở nên quan trọng.

Ứng dụng điện thoại thông minh mang đến một cuộc cách mạng mới trong công nghệ kiểm thử phần mềm. Bởi kiểm thử trên điện thoại di động thoát khỏi kiến trúc của máy tính về phân cứng và hoàn toàn khác trong cách người dùng tương tác. Các phần mềm di động cung cấp các thao tác trượt, chạm và nhấp của người dùng nên làm thế nào để đạt được việc thực thi kiểm thử tự động trên các máy di động đã trở thành một khó khăn. Ngoài ra, do sự tồn tại phổ biến của cả hệ điều hành IOS và Android, hầu hết các ứng dụng di động cần hỗ trợ nhiều nền tảng, điều này khiến framework kiểm thử di động tự động có yêu cầu cao hơn về tính tương thích.

Giải quyết các vấn đề trên, Appium mang đến những giải pháp hữu ích, giúp nâng cao hiệu quả việc kiểm thử ứng dụng di động tự động và giúp doanh nghiệp tiết kiệm chi phí.

Để tìm hiểu về công cụ này, bản báo cáo có ba phần chính như sau

- Phần 1: Giới thiệu về Appium, kiến trúc và cùng phân tích ưu nhược điểm của công cụ. Từ đó có một số so sánh với các công cụ kiểm thử di động tự động hiện nay.
- Phần 2: Trình bày các yêu cầu cài đặt để viết và thực thi các ca kiểm thử tự động sử dụng Appium.
- Phần 3: Thực hiện viết, mô tả các ca kiểm thử cho một số chức năng của một dự án mã nguồn mở. Từ đó đưa ra kết luận.

Kết thúc, người đọc sẽ có thể hiểu được cách Appium hoạt động, lợi thế cũng như cách sử dụng để có thể cân nhắc dùng Appium cho việc kiểm thử tự động các ứng dụng di động sau này.

Mục lục

Giới thiệu	ii
Mục lục	iii
Danh mục hình ảnh	v
Danh mục bảng biểu.....	vii
1. Giới thiệu Appium.....	1
1.1. Appium là gì?	1
1.1.1. Phân loại ứng dụng di động:	1
1.1.2. Đặc trưng	1
1.2. Một số khái niệm.....	2
1.2.1. Appium Server	2
1.2.2. Appium Client.....	2
1.2.3. JSON Wire Protocol	2
1.2.4. Appium Driver	3
1.2.5. Automated Testing Framework	3
1.2.6. Appium Session	3
1.2.7. Desired capabilities.....	3
1.3. Kiến trúc Appium.....	4
1.4. Tại sao chọn Appium?	6
1.5. Nhược điểm.....	7
2. Cài đặt.....	8
2.1. Yêu cầu về hệ thống	8
2.1.1. Android	8
2.1.2. IOS	8
2.2. Cài đặt Appium:	8
2.2.1. Cài đặt Appium Server bằng Command Prompt	8
2.2.2. Cài đặt GUI-based Appium Server.....	11
2.3. Cài đặt trình điều khiển.....	13

2.4. Appium Inspector.....	15
2.5. Appium Client.....	19
2.6. Appium Grid	19
3. Kiểm thử ứng dụng.....	21
3.1. Ứng dụng Telegram	21
3.2. Chức năng	22
3.3. Các ca kiểm thử.....	23
3.3.1. Các ca kiểm thử cho chức năng gửi tin nhắn.....	25
3.3.2. Các ca kiểm thử cho chức năng cài đặt thông tin, giao diện người dùng...31	31
3.3.3. Thực thi các ca kiểm thử trên nền tảng Android	34
3.3.4. Thực thi ca kiểm thử trên nền tảng Windows:.....	34
3.4. Kết quả kiểm thử:	36
3.5. Kiểm thử trên nhiều nền tảng với Appium Grid:	36
4. Tổng kết	40
5. Tài liệu tham khảo.....	40

Danh mục hình ảnh

Hình 1.1. Kiến trúc Appium	4
Hình 1.2. Kiến trúc Appium trên Android	5
Hình 1.3. Kiến trúc Appium trên IOS	5
Hình 2.1. Cài đặt Nodejs và npm (1).....	9
Hình 2.2. Cài đặt Nodejs và npm (2).....	9
Hình 2.3. Cài đặt Nodejs và npm (3).....	10
Hình 2.4. Cài đặt và bắt đầu Appium Server qua Command Prompt.....	11
Hình 2.5. Giao diện Appium Desktop	12
Hình 2.6. Appium Desktop: Khởi chạy máy chủ lắng nghe kết nối.	13
Hình 2.7. Cài đặt Android SDK qua Android Studio (1)	14
Hình 2.8. Cài đặt Android SDK (2)	14
Hình 2.9. Cài đặt Android SDK (3)	15
Hình 2.10. Cài đặt Appium Inspector.....	16
Hình 2.11. Giao diện Appium Inspector.....	16
Hình 2.12. Appium Inspector: Thiết lập các giá trị cho Desired Capabilities.....	17
Hình 2.13. Appium Inspector: Bắt đầu phiên.....	18
Hình 2.14. Appium Inspector: Chức năng Select Elements	18
Hình 2.15. Appium Inspector: Chức năng Recording	19
Hình 2.16. IntelliJ: Tạo Maven Project viết các tập lệnh sử dụng ngôn ngữ Java	19
Hình 2.17. Kiến trúc Appium Grid	20
Hình 2.18. Selenium Grid Console	20
Hình 3.1. Cấu trúc mã nguồn của các tập lệnh kiểm thử	24
Hình 3.2. MobileTest: Tìm kiếm người dùng và truy cập tin nhắn.....	25
Hình 3.3. MobileTest: Nhắn tin văn bản	26
Hình 3.4. MobileTest: Nhắn biểu tượng cảm xúc.....	27
Hình 3.5. MobileTest: Gửi nhãn dán	27
Hình 3.6. MobileTest: Gửi hình ảnh.....	28
Hình 3.7. MobileTest: Chính sửa ảnh trước khi gửi	29

Hình 3.8. MobileTest: Chia sẻ địa chỉ.....	30
Hình 3.9. MobileTest: Xóa một cuộc trò chuyện	30
Hình 3.10. MobileTest: Xóa nhiều cuộc trò chuyện	31
Hình 3.11. MobileTest: Thay đổi giao diện	32
Hình 3.12. BeforeMethod: Mở mục quản lý thông tin tài khoản.....	32
Hình 3.13. MobileTest: Thay đổi tên người dùng	33
Hình 3.14. MobileTest: Thay đổi tiểu sử người dùng	34
Hình 3.15. Thiết lập để kiểm thử trên Android Device.....	34
Hình 3.16. Kiểm thử trên Window Desktop (1).....	35
Hình 3.17. Kiểm thử trên Window Desktop (2): Thay đổi giao diện người dùng .	35
Hình 3.18. Kết quả thực thi trên Android Device	36
Hình 3.19. Kết quả thực thi trên Windows Desktop	36
Hình 3.20. Appium Grid: Tệp cấu hình cho thiết bị 1	37
Hình 3.21. Appium Grid: Tệp cấu hình cho thiết bị 2	37
Hình 3.22. Testcase cho kiểm thử song song (1)	39
Hình 3.23. Testcase cho kiểm thử song song (2)	39
Hình 3.24. Kết quả kiểm thử song song.....	39

Danh mục bảng biểu

Bảng 1.1. So sánh các công cụ kiểm thử di động tự động theo bốn triết lý Appium	7
---	---

1. Giới thiệu Appium

1.1. Appium là gì?

Appium là một công cụ kiểm thử tự động mã nguồn mở, tập trung hướng tới các ứng dụng di động IOS/Android, ngoài ra cũng hỗ trợ kiểm thử các ứng dụng trên Window Desktop. Bản phát hành ban đầu năm 2011 được phát triển bởi Dan Cuellar, viết bằng ngôn ngữ lập trình C#. Đến năm 2013, khi được tài trợ và duy trì với Sauce Labs, Jonathan Lipps cùng với nhóm của mình đã sử dụng Node.js làm framework để phục dựng lại Appium và xác định JavaScript là ngôn ngữ có thể đưa Appium phát triển thành một cộng đồng mã nguồn mở lớn hơn.

1.1.1. Phân loại ứng dụng di động:

Các ứng dụng di động được chia thành ba loại: Native, Web và Hybrid app

- **Native apps:** là các ứng dụng gốc được phát triển cho một nền tảng cụ thể với một ngôn ngữ lập trình cụ thể (ví dụ Objective-C cho IOS hay Java cho Android), được cài đặt trực tiếp vào thiết bị thông qua cửa hàng ứng dụng (Google Play hay App Store). Các ứng dụng này có thể có quyền sử dụng các ứng dụng hệ thống - ứng dụng mặc định của thiết bị di động như camera, GPS, ...
- **Web apps:** là các ứng dụng được truy cập trên thiết bị di động thông qua các trình duyệt web (Appium hỗ trợ Safari trên IOS; Chrome hoặc các ứng dụng trình duyệt tích hợp trên Android) và sẽ thích ứng với bất kỳ thiết bị nào mà người dùng sử dụng. Các ứng dụng mobile web thì không cần tải và cài đặt trên thiết bị.
- **Hybrid apps:** là ứng dụng web được nhúng trong ứng dụng native, được phát triển bằng các công nghệ web (html, css, js) - có thể được hiểu như là sự kết hợp của Web app và Native app.

1.1.2. Đặc trưng

Một số đặc trưng nổi bật của Appium:

- **Hỗ trợ đa ngôn ngữ, đa nền tảng:** bởi cho phép viết các câu lệnh kiểm thử tự động bằng nhiều ngôn ngữ lập trình, như Java, PHP, Python, Perl, ..., và thực thi chúng trên nhiều nền tảng khác nhau (IOS, Android hay Windows) với cùng API.

- **JSON Wire Protocol:** Appium sử dụng cơ chế truyền tải theo giao thức JSON Wire Protocol được tạo bởi các nhà phát triển Selenium WebDriver nhằm mục đích kiểm thử tự động hóa trình duyệt web.
- **Không cần biên dịch lại:** không cần biên dịch lại sau mỗi thay đổi nhỏ như công cụ kiểm thử di động Robotium, do đó tiết kiệm được thời gian.
- **Hỗ trợ kiểm thử trên cả thiết bị vật lý và các thiết bị giả lập:** Có thể kiểm thử hiệu quả trên trình giả lập, như các trường hợp cần kiểm tra độ chịu tải khi có hàng nghìn lượt truy cập, ngoài ra còn có thể kiểm thử trên các thiết bị thật, phù hợp với các trường hợp hoạt động của ứng dụng khi bị gián đoạn bởi một tin nhắn hay cuộc gọi, ...

1.2. Một số khái niệm

1.2.1. Appium Server

Appium Server là một máy chủ HTTP được viết bằng Node.js, thực hiện nhận các yêu cầu HTTP từ máy khách qua giao thức JSON và xử lý các yêu cầu đó theo nhiều cách, phụ thuộc vào nền tảng kiểm thử. Kết quả kiểm thử sẽ được nhận và gửi lại cho máy khách.

Appium Server có khả năng tạo nhiều phiên để thực hiện kiểm thử đồng thời trên nhiều thiết bị khác nhau.

1.2.2. Appium Client

Là các thư viện máy khách hỗ trợ cho giao thức WebDriver, gồm đầy đủ các ngôn ngữ phổ biến Java, Python, Ruby, PHP, JavaScript và C#.

1.2.3. JSON Wire Protocol

Đây là một giao thức tạo điều kiện giao tiếp giữa các thư viện máy khách và máy chủ trong hệ thống không đồng nhất, gồm tập các điểm đầu cuối tiêu chuẩn được định nghĩa trước thông qua API RESTful.

Trong kiến trúc Appium, giao tiếp giữa thư viện máy khách và máy chủ Appium được hỗ trợ bởi WebDriver sử dụng JSON Wire Protocol. Do đó, máy chủ không cần hiểu ngôn ngữ lập trình của các thư viện máy khách mà thay vào đó, nó chỉ cần xác định giao thức để tạo và quản lý các phiên kiểm thử tự động.

Kiến trúc của giao thức dây JSON:

- Local End: phía client, ở đây là các thư viện máy khách Appium được viết bởi một ngôn ngữ cụ thể
- Remote End: phía server, đọc các yêu cầu từ Local End và gửi lại phản hồi thông qua một TCP socket

- Intermediary End: là hệ thống proxy có thể hoạt động như một Remote End hay như một Local End
- Endpoint Node: Remote End cuối cùng trong cấu trúc node.

1.2.4. Appium Driver

Để quản lý thực thi tập lệnh kiểm thử trên các nền tảng khác nhau một cách hiệu quả, Appium sử dụng đa dạng trình điều khiển và chuyển đổi API thành các phiên tự động cho các nền tảng tương ứng. Một số trình điều khiển được sử dụng bởi Appium:

- UIAutomator1/2
- XCUIITest
- WinAppDriver
- Espresso

1.2.5. Automated Testing Framework

Appium sử dụng các framework kiểm thử tự động để thực thi các câu lệnh nhận từ máy khách trên thiết bị kiểm thử.

- Với IOS: Apple's XCUIITest cho IOS 9.3+, Apple's UIAutomation cho các phiên bản thấp hơn
- Với Android: Google's UIAutomator/UIAutomator 2 cho các phiên bản 4.3+ với SDK >16

1.2.6. Appium Session

Tự động hóa luôn được thực hiện trong ngữ cảnh của một phiên. Thư viện máy khách sẽ yêu cầu máy chủ tạo một phiên và nhận phản hồi từ máy chủ một mã định danh phiên sessionId. Máy khách sẽ sử dụng mã định danh này cho việc gửi các câu lệnh khác để tương tác với ứng dụng cần kiểm thử.

1.2.7. Desired capabilities

Desired capabilities là một đối tượng JSON được gửi bởi máy khách tới máy chủ Appium, miêu tả các khả năng mà chúng ta mong muốn cho phiên kiểm thử tự động, một số khả năng:

- *platformName*: để xác định hệ điều hành IOS, Android hay Windows
- *deviceName*: xác định tên của thiết bị di động được sử dụng cho hoạt động kiểm thử ứng dụng.
- *app*: đường dẫn tuyệt đối hoặc địa chỉ URL tới file ứng dụng cần kiểm thử (có thể là .ipa cho IOS, .apk cho Android, .app cho IOS Simulator hay .zip) để

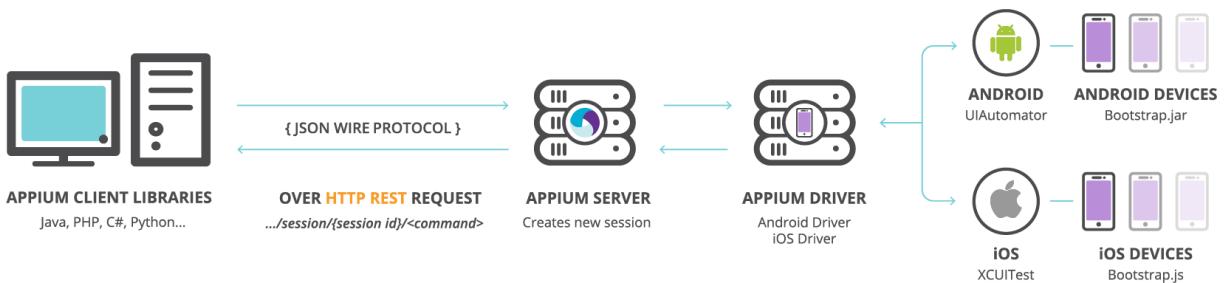
Appium cài đặt vào thiết bị. Với các ứng dụng đã có sẵn trên thiết bị Android, app sẽ được thay thế bởi *appPackage* và *appActivity*.

- *browserName*: với các web apps, cần có trường *browserName* để xác định trình duyệt.
- *noReset*: được sử dụng để khởi động lại trạng thái của ứng dụng trước khi phiên kiểm thử được bắt đầu (với giá trị bằng false), ngược lại, để giữ nguyên trạng thái trước đó của ứng dụng, giá trị được đặt bằng true.

Có một số Desired capabilities chỉ áp dụng cho một nền tảng hệ điều hành hay một trình điều khiển cụ thể, ví dụ như *appPackage* và *appActivity* chỉ dành cho Android; *appName* cho iOS, ...

1.3. Kiến trúc Appium

Appium được phát triển dựa trên Selenium – một giao thức của Google để tự động hóa kiểm thử web trên các trình duyệt, do đó kế thừa kiến trúc máy khách – máy chủ từ Selenium.



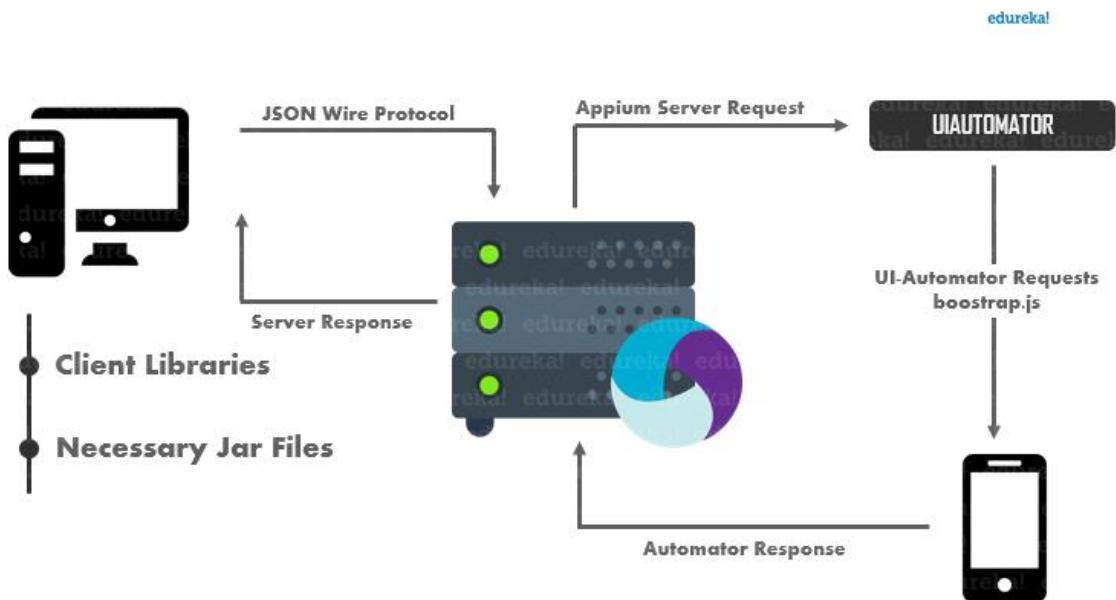
Hình 1.1. Kiến trúc Appium

Appium nhận kết nối từ máy khách, lắng nghe lệnh thông qua các yêu cầu HTTP REST sau đó thực thi các lệnh đó trên thiết bị di động và phản hồi bằng các phản hồi HTTP đại diện cho kết quả thực hiện.

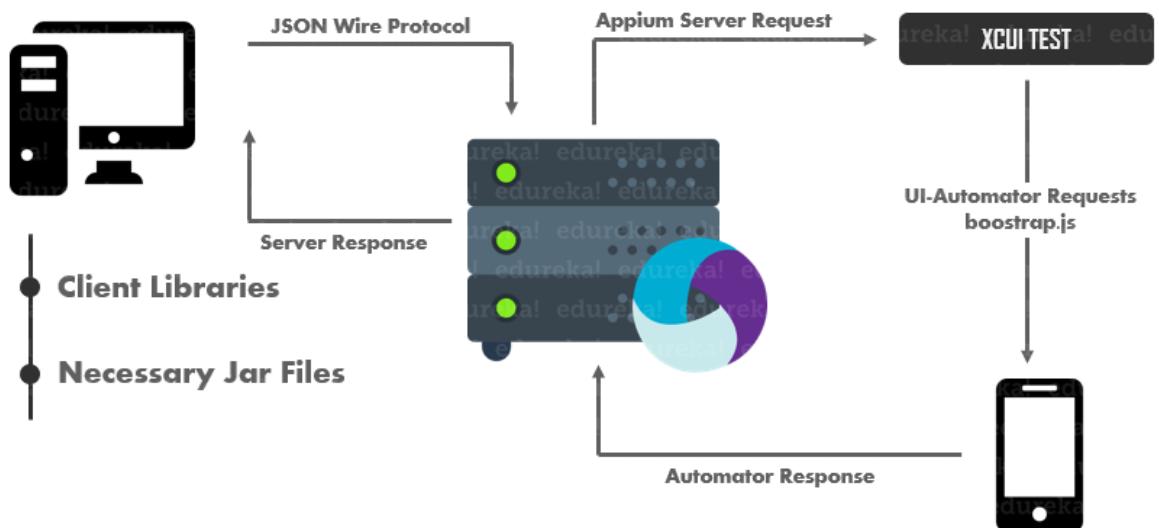
Để bắt đầu một phiên kiểm thử tự động, máy khách sẽ gửi một yêu cầu tới Appium Server thông qua giao thức dây JSON (JSONWP). Tập lệnh kiểm thử có thể được viết dưới nhiều ngôn ngữ lập trình phổ biến như Java, Python, ... Appium Server sẽ tạo một phiên kiểm thử mới và gọi đến một trình điều khiển thích hợp (Android/iOS Driver). Sau khi kết nối tới một framework kiểm thử tương ứng, Appium Server sẽ bắt đầu tương tác với dịch vụ bootstrap (bootstrap.js với iOS và bootstrap.jar với Android) đang chạy trên thiết bị di động để thực hiện các hoạt động giống như người dùng.

Appium cho phép dùng cùng một tập lệnh để kiểm tra các thiết bị iOS và Android. Sự khác nhau giữa yêu cầu kiểm thử tự động giữa hai nền tảng này sẽ được xử lý bởi

Appium qua các sự lựa chọn trong Desired Capabilities. Do đó, trước khi tạo một phiên, cần phải thiết lập các giá trị như platformName để Appium tạo phiên kiểm thử phù hợp.



Hình 1.2. Kiến trúc Appium trên Android



Hình 1.3. Kiến trúc Appium trên IOS

Trên hai nền tảng IOS/Android, sự khác biệt được xác định khi Appium Server xem xét Desired Capabilities. Với nền tảng Android, máy chủ sẽ sử dụng framework kiểm thử tự động cho Android, mặc định là UIAutomator2; với nền tảng IOS, máy chủ sử dụng framework XCUI Test. Các framework này sẽ tương tác với file bootstrap trên thiết bị (hoạt động như một TCP server). File bootstrap nhận câu lệnh từ framework và

thực thi trên thiết bị, sau đó sẽ gửi lại chi tiết test log. Những thông tin này sẽ được gửi lại cho máy khách.

1.4. Tại sao chọn Appium?

Appium được thiết kế để đáp ứng nhu cầu kiểm thử tự động hóa trên các thiết bị di động, tuân theo 4 triết lý, cũng như là các ưu điểm sau:

- 1) Không cần phải biên dịch lại hay sửa đổi ứng dụng để có thể tự động hóa kiểm thử.**

Appium sử dụng các framework kiểm thử tự động có sẵn giúp chúng ta không cần phải bổ sung và biên dịch bất cứ đoạn mã hoặc framework nào dành riêng cho Appium hoặc bên thứ ba trong ứng dụng cần kiểm thử. Điều đó có nghĩa là bạn đang kiểm thử chính xác ứng dụng được phát hành tới tay người dùng.

Hiện nay có nhiều công cụ kiểm thử tự động di động mã nguồn mở (như Robotium) cần biên dịch lại hoặc thay đổi mã nguồn ứng dụng trước khi thực thi kiểm thử và thậm chí sau khi kiểm thử, chúng ta cần loại bỏ những phần thay đổi đó. Tức trong trường hợp này, chúng ta đang không thực sự kiểm thử ứng dụng được phát hành trên cửa hàng ứng dụng.

- 2) Không bị giới hạn bởi một ngôn ngữ lập trình hay một framework cụ thể để viết và thực thi các ca kiểm thử.**

Appium mở rộng các thư viện máy khách Selenium WebDriver - đã được viết bằng các ngôn ngữ lập trình phổ biến như Java, Objective-C, JavaScript, PHP, Python, Ruby, C#, Clojure hay Perl. Vậy nên chúng ta có thể sử dụng bất kể ngôn ngữ lập trình nào trong số đó để viết các tập lệnh kiểm thử tự động.

Các thư viện máy khách chỉ đơn giản là máy khách HTTP và được kết hợp vào trong mã nguồn tập lệnh kiểm thử. Nói cách khác, các máy khách Appium và WebDriver không phải là một framework kiểm thử mà là các thư viện tự động hóa. Người kiểm thử có thể quản lý môi trường thử nghiệm theo bất kỳ cách nào họ muốn.

Nếu bạn sử dụng thư viện XCUITest của Apple mà không có Appium, bạn chỉ có thể viết các tập lệnh kiểm thử bằng Objective – C hoặc Swift và chỉ có thể chạy các tập lệnh qua Xcode. Tương tự với UIAutomator hoặc Espresso của Google, bạn chỉ có thể viết bằng Java hoặc Kotlin. Appium đã mở ra khả năng tự động hóa ứng dụng đa nền tảng thực sự.

- 3) Một framework tự động hóa di động không nên phát minh lại các APIs tự động.**

Selenium WebDriver đã trở thành tiêu chuẩn để tự động hóa các trình duyệt web. Câu hỏi đặt ra là tại sao phải làm điều gì đó hoàn toàn khác biệt cho các thiết bị di động? Thay vào đó, Appium đã mở rộng giao thức Selenium JSONWP với việc bổ sung thêm các phương thức API phù hợp. Vậy nên nó có cùng tiêu chuẩn với Selenium WebDriver và không cần phải phát minh lại framework kiểm thử di động tự động.

4) Một công cụ kiểm thử tự động nên là mã nguồn mở.

Rõ ràng, appium là một công cụ mã nguồn mở, có thể dễ dàng tìm thấy trên github.com/appium/appium và các thông tin chi tiết về công cụ này đều được công khai trên appium.io

Những ưu điểm trên đã giúp cho Appium vượt trội hơn so với các công cụ tự động hóa di động khác, được thể hiện rõ qua sự so sánh dưới đây:

Bảng 1.1. So sánh các công cụ kiểm thử di động tự động theo bốn triết lý Appium

Công cụ	Triết lý 1	Triết lý 2	Triết lý 3	Triết lý 4
Appium	Có	Có	Có	Có
IOS Driver	Có	Có	Có	Không
Robotium	Không	Không	Có	Không
Selendroid	Không	Có	Có	Không
Calabash	Không	Không	Không	Có

1.5. Nhược điểm

Bên cạnh những lợi thế vượt trội trên, Appium vẫn có một số điểm yếu:

- Giới hạn hỗ trợ cho các phiên bản Android: Appium chỉ hỗ trợ trực tiếp cho các phiên bản Android 17 hoặc hơn, vậy nên để hỗ trợ các API cũ hơn, cần phải sử dụng thư viện mã nguồn mở khác như Selendroid.
- Giới hạn trong kiểm thử Hybrid App: không thể kiểm thử được hành động chuyển đổi từ web sang native app và ngược lại.
- Appium Inspector không hỗ trợ cho tìm kiếm vị trí của các phần tử ứng dụng trên Window Desktop.
- Hạn chế kỹ thuật ở IOS khi mà chỉ có thể chạy tập lệnh kiểm thử IOS trên một thiết bị cho mỗi máy MAC, vậy nên muốn thực thi ở nhiều thiết bị IOS cùng

lúc thì cần nhiều máy MAC. Hạn chế này có thể giải quyết khi thực thi các tập lệnh ở các nền tảng dịch vụ cloud như Sauce Labs hay LambdaTest.

2. Cài đặt

2.1. Yêu cầu về hệ thống

2.1.1. Android

- Java (phiên bản 7 trở lên)
- Android SDK (phiên bản 17 trở lên)
- Android Virtual Device (AVD) - Emulator hoặc thiết bị di động Android
- IDE để viết các tập lệnh, ở đây nhóm chọn ngôn ngữ Java với IDE IntelliJ
- TestNG, thư viện máy khách Appium (Java-client), thư viện Selenium Server and WebDriver Java
- Appium Server
- Ứng dụng cài đặt sẵn trên thiết bị hoặc file APK của ứng dụng.

2.1.2. IOS

- MAC OS X 10.7 trở lên
- Xcode
- Java (phiên bản 7 trở lên)
- Simulator hoặc thiết bị di động IOS
- IDE
- TestNG, thư viện máy khách Appium , Selenium Webdriver Java Library
- Appium Server

2.2. Cài đặt Appium:

Appium có hai phiên bản:

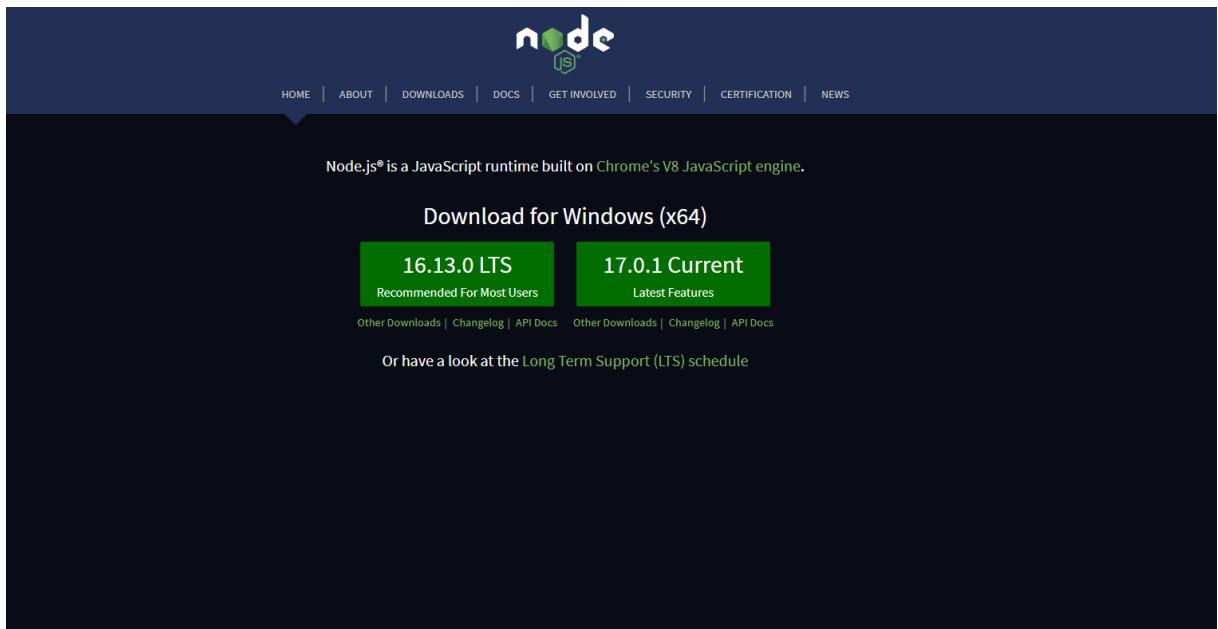
- CLI-based Appium Server
- GUI-based Appium Desktop:

2.2.1. Cài đặt Appium Server bằng Command Prompt

Với phiên bản CLI-based Appium Server – phiên bản chạy bằng lệnh của Appium, chúng ta sẽ cài đặt thông qua Command Prompt. Để có thể thực hiện, cần đến sự hỗ trợ của Nodejs và package quản lý NPM.

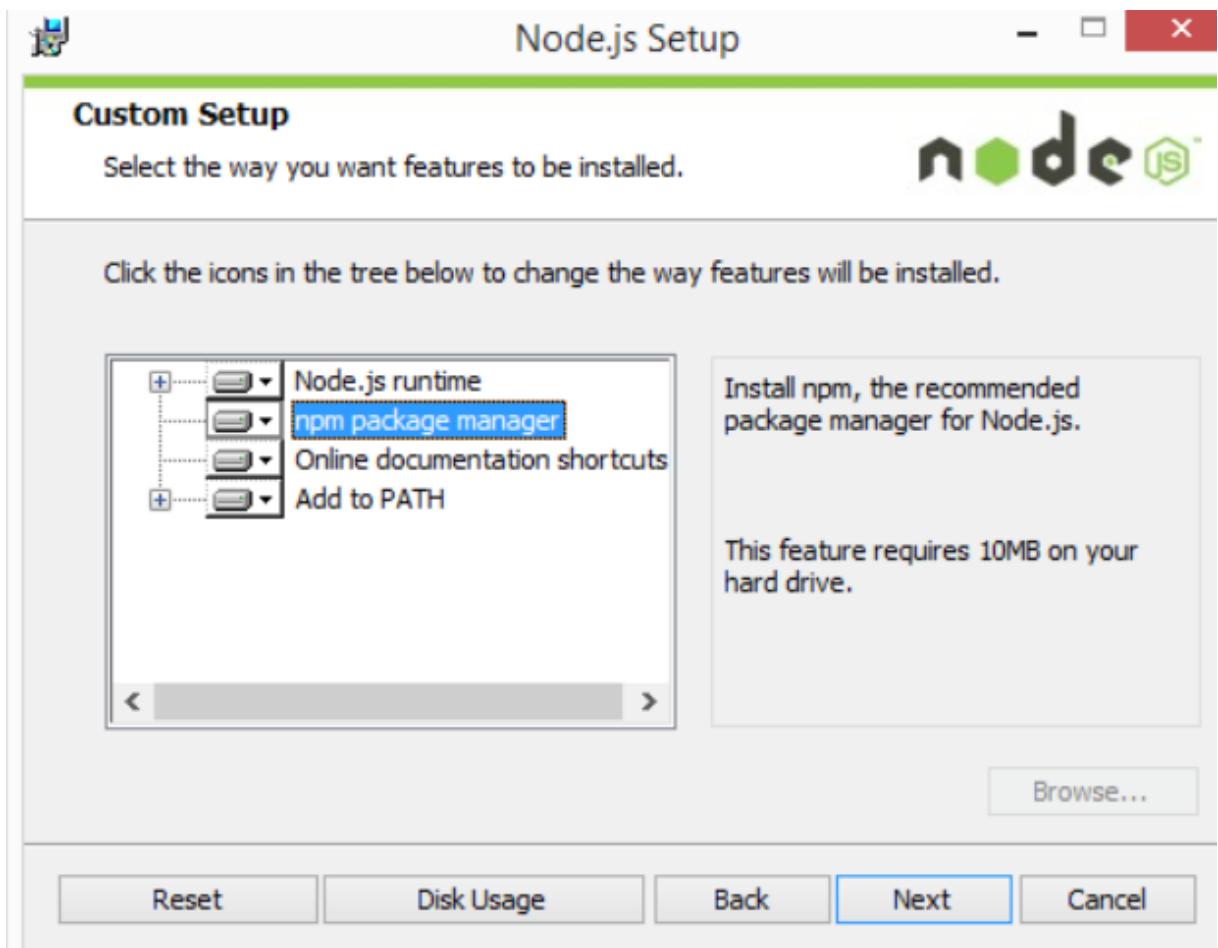
Các bước cài đặt:

- 1) Cài đặt Node.js trên [Nodejs.org](https://nodejs.org)



Hình 2.1. Cài đặt Nodejs và npm (1)

- 2) Chạy file .msi vừa tải và làm theo các chỉ dẫn cài đặt (chọn cài đặt npm package manager)

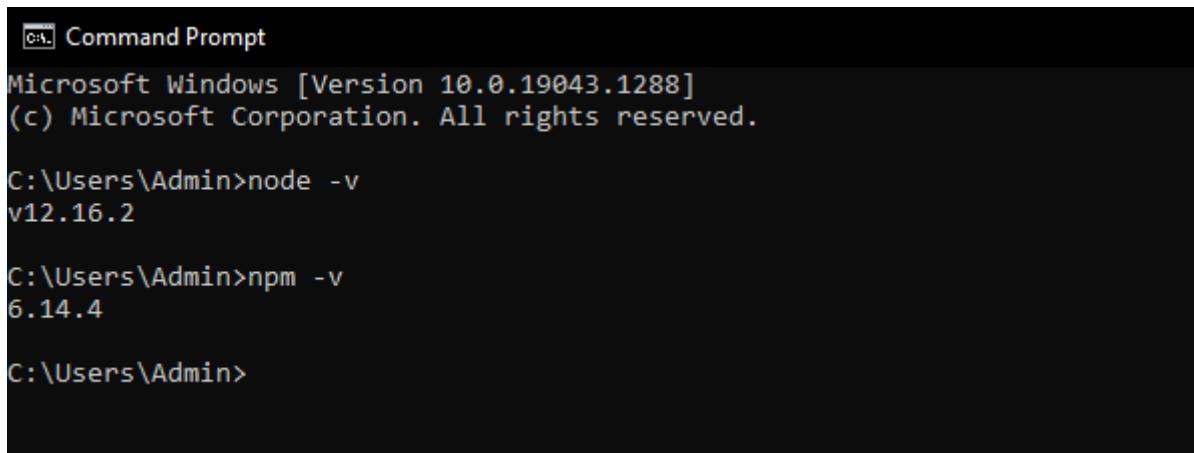


Hình 2.2. Cài đặt Nodejs và npm (2)

3) Khởi động lại máy tính: kiểm tra cài đặt bằng việc mở Command Prompt, gõ lệnh

➤ ***node -v***

➤ ***npm -v***



```
Command Prompt
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>node -v
v12.16.2

C:\Users\Admin>npm -v
6.14.4

C:\Users\Admin>
```

Hình 2.3. Cài đặt Nodejs và npm (3)

Nếu cài đặt thành công, phiên bản của Nodejs và npm được hiện trên màn hình.

4) Cài đặt Appium:

➤ ***npm install -g Appium***

Khi đó npm sẽ thực hiện tải xuống Appium

Để bắt đầu Appium Server, thực hiện câu lệnh:

➤ ***appium -a 127.0.0.1 -p 4723***

Giờ đây, Appium đang được chạy và REST HTTP đang lắng nghe dưới địa chỉ ip localhost và cổng 4723.

```

C:\ Node.js command prompt - appium -a 127.0.0.1 -p 4723
C:\Users\Admin>npm install -g appium
C:\Users\Admin\AppData\Roaming\npm\appium -> C:\Users\Admin\AppData\Roaming\npm\node_modules\appium\build\lib\main.js
> appium-windows-driver@1.19.1 install C:\Users\Admin\AppData\Roaming\npm\node_modules\appium\node_modules\appium-windows-driver
> node install-npm.js

Info: WinAppDriver WinAppDriver doesn't exist, setting up
You are not running as an administrator so WinAppDriver cannot be installed for you; please reinstall as admin
WinAppDriver was not installed; please check your system and re-run npm install if you need WinAppDriver

> core-js@2.6.12 postinstall C:\Users\Admin\AppData\Roaming\npm\node_modules\appium\node_modules\core-js
> node -e "try{require('./postinstall')}catch(e){}"

Thank you for using core-js ( https://github.com/zloirock/core-js ) for polyfilling JavaScript standard library!

The project needs your help! Please consider supporting of core-js on Open Collective or Patreon:
> https://opencollective.com/core-js
> https://www.patreon.com/zloirock

Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)

> appium-chromedriver@4.27.3 postinstall C:\Users\Admin\AppData\Roaming\npm\node_modules\appium\node_modules\appium-chromedriver
> node install-npm.js

dbug ChromedriverStorageClient Parsed 510 entries from storage XML
Info: ChromedriverStorageClient The total count of entries in the mapping: 362
dbug ChromedriverStorageClient Selecting chromedrivers whose versions match to 91.0.4472.101
dbug ChromedriverStorageClient Got 4 items
dbug ChromedriverStorageClient Selecting chromedrivers whose platform matches to win32
dbug ChromedriverStorageClient Got 1 item
dbug ChromedriverStorageClient Got 1 driver to sync: [
dbug ChromedriverStorageClient "91.0.4472.101/chromedriver_win32.zip"
dbug ChromedriverStorageClient ]
dbug ChromedriverStorageClient Retrieving 'https://chromedriver.storage.googleapis.com/91.0.4472.101/chromedriver_win32.zip' to 'C:\Users\Admin\AppData\Local\Temp\2021103-13420-1idyjsy.m3sug\0.zip'
dbug Support Traversed 1 directory and 1 file in 2ms
dbug ChromedriverStorageClient Moving the extracted 'chromedriver.exe' to 'C:\Users\Admin\AppData\Roaming\npm\node_modules\appium\node_modules\appium-chromedriver\chromedriver\chromedriver\win\chromedriver_win32_v91.0.4472.101.exe'
dbug ChromedriverStorageClient Permissions of the file 'C:\Users\Admin\AppData\Roaming\npm\node_modules\appium\node_modules\appium-chromedriver\chromedriver\chromedriver_win32_v91.0.4472.101.exe' have been changed to 755
Info: ChromedriverStorageClient Successfully synchronized 1 chromedriver
+ appium@1.22.0
added 676 packages from 640 contributors in 35.518s

C:\Users\Admin>appium --version
1.22.0

C:\Users\Admin>appium -a 127.0.0.1 -p 4723
[Appium] Welcome to Appium v1.22.0
[Appium] Non-default server args:
[Appium]   address: 127.0.0.1
[Appium] Appium REST http interface listener started on 127.0.0.1:4723

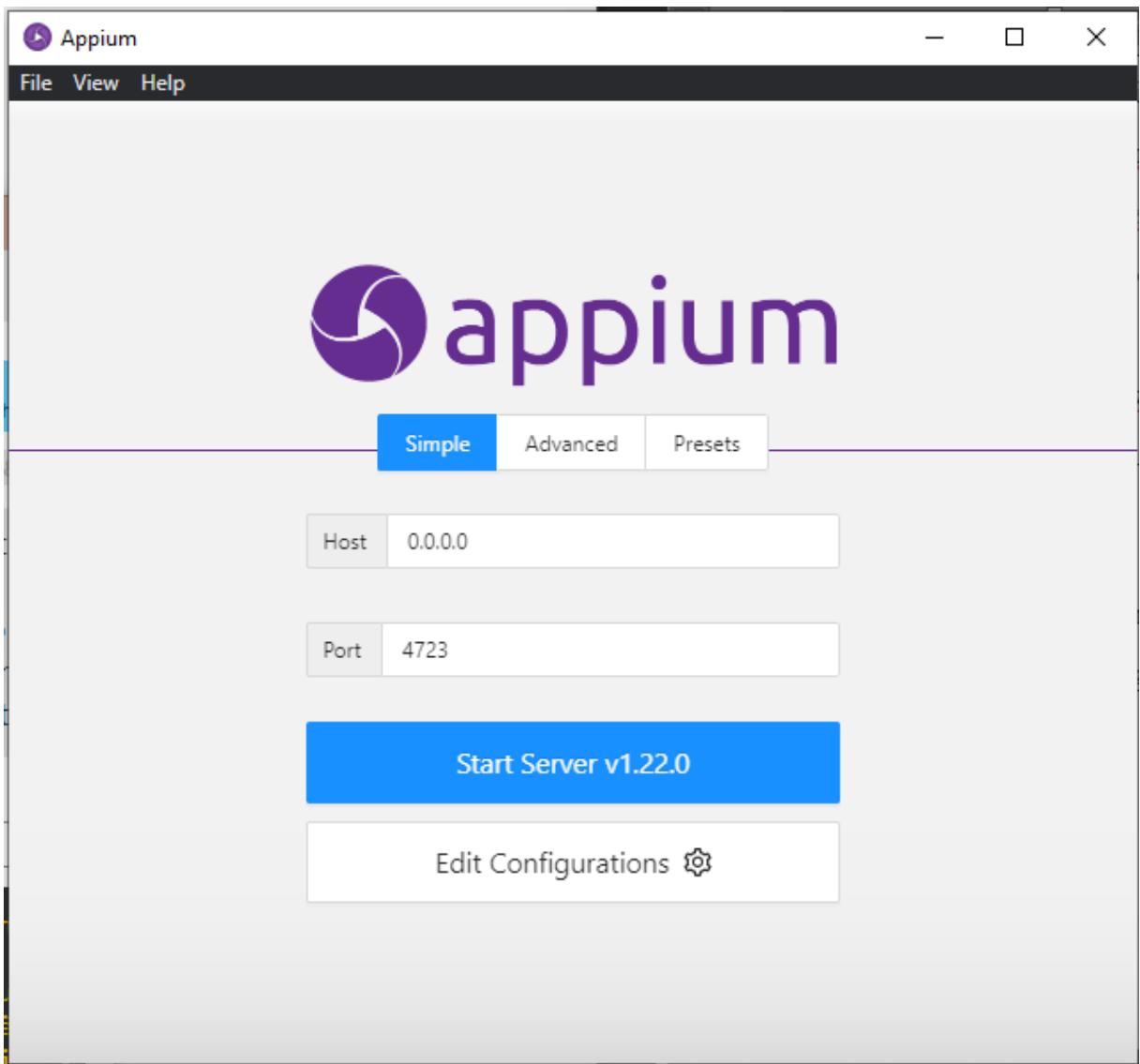
```

Hình 2.4. Cài đặt và bắt đầu Appium Server qua Command Prompt

2.2.2. Cài đặt GUI-based Appium Server

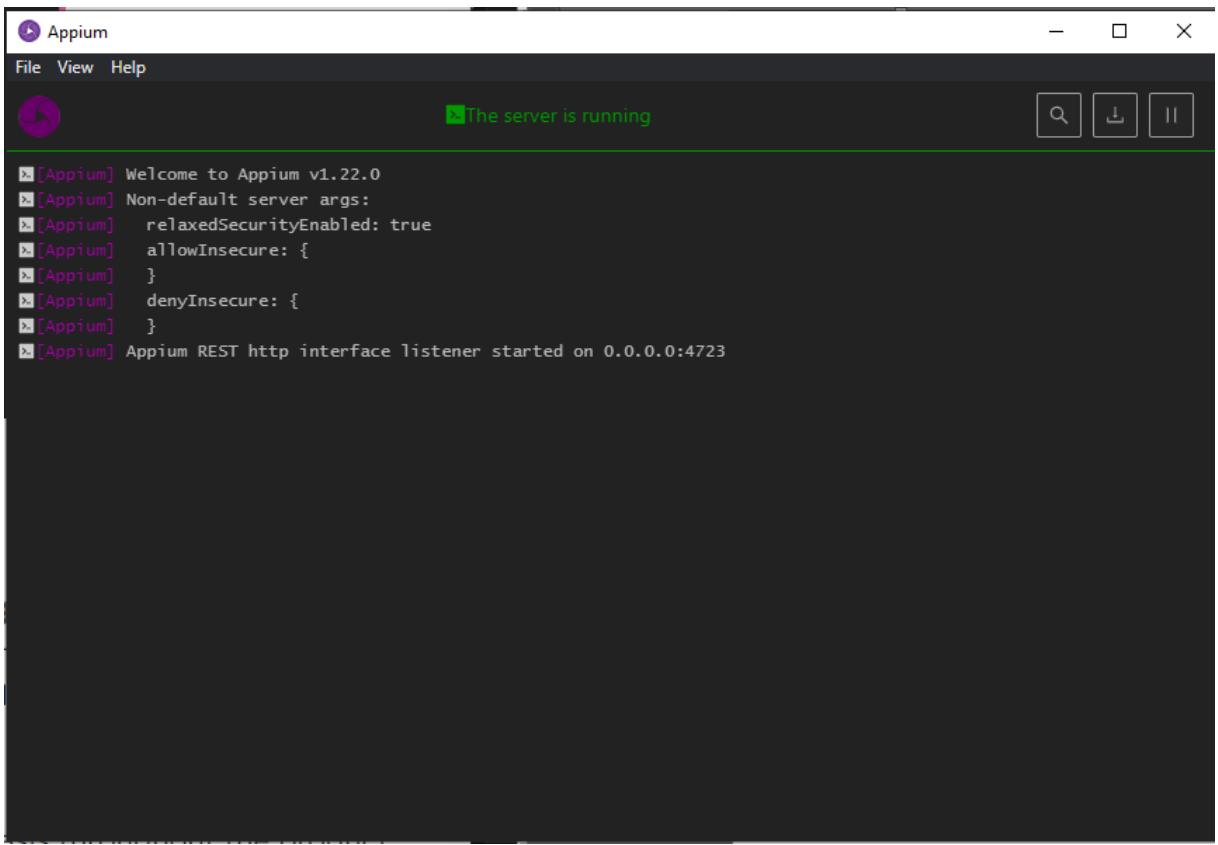
Appium cung cấp Appium Desktop - phiên bản ứng dụng cho máy chủ Appium với giao diện người dùng thân thiện, dễ sử dụng và có thể được tải xuống cho bất kỳ nền tảng nào. Appium Desktop đi kèm với mọi thứ cần thiết để chạy máy chủ Appium, do đó, không cần phải tải Node.js và npm theo cách tiếp cận bằng Command Prompt như trên.

Thực hiện tải Appium Desktop từ trang chính thức của Appium appium.io



Hình 2.5. Giao diện Appium Desktop

Ở giao diện trang đầu, Appium yêu cầu nhập IP và cổng của server, mặc định là localhost với cổng 4723. Chọn Start Server để bắt đầu máy chủ Appium.

A screenshot of the Appium Desktop application window. The title bar says "Appium". The menu bar includes "File", "View", and "Help". The main area shows command-line output:

```
[Appium] Welcome to Appium v1.22.0
[Appium] Non-default server args:
[Appium]   relaxedSecurityEnabled: true
[Appium]   allowInsecure: {
[Appium]     }
[Appium]   denyInsecure: {
[Appium]     }
[Appium] Appium REST http interface listener started on 0.0.0.0:4723
```

At the top right, there is a status message "The server is running" with a green checkmark icon. To the right of the message are three small icons: a magnifying glass for search, a downward arrow for refresh, and a double vertical line for pause.

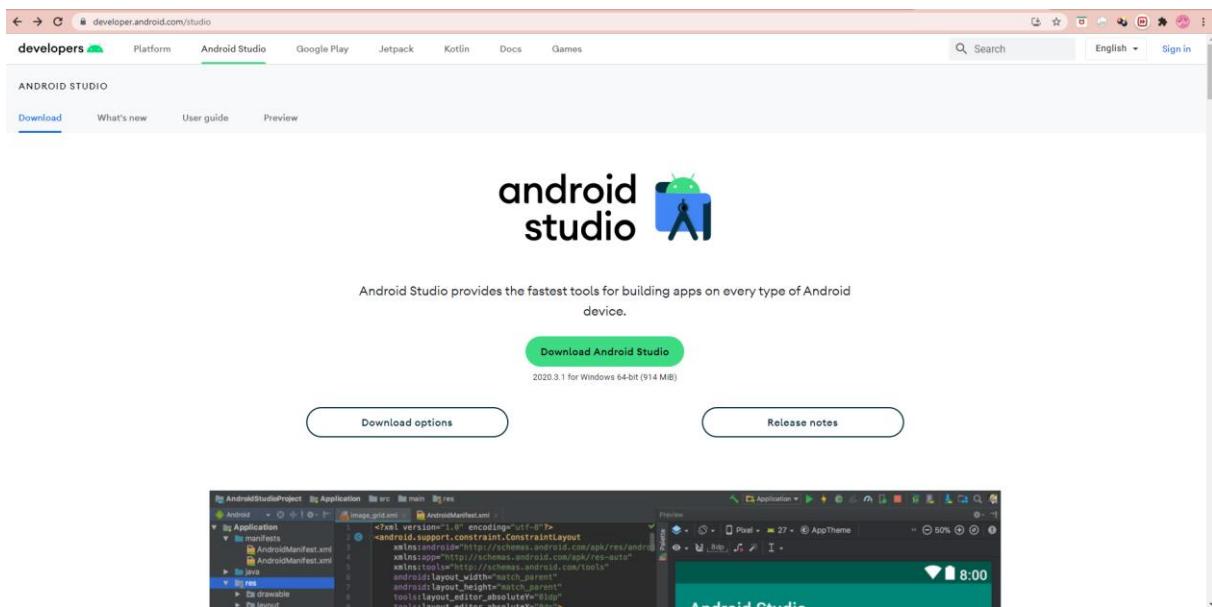
Hình 2.6. Appium Desktop: Khởi chạy máy chủ lắng nghe kết nối.

2.3. Cài đặt trình điều khiển

Appium hỗ trợ tự động hóa trên các nền tảng khác nhau nhờ các trình điều khiển Appium tương ứng. Để tự động hóa các ứng dụng Android bằng một trong các trình điều khiển Appium cung cấp, cần phải có Android SDK đã được cấu hình trên hệ thống.

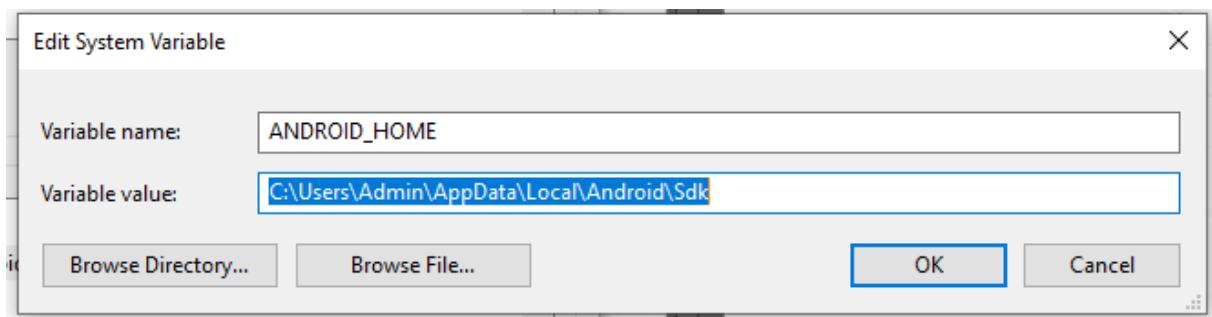
Ở hướng dẫn này, nhóm chọn cài đặt Android Studio – bao gồm đầy đủ các SDK tool packages, và đồng thời hỗ trợ nhóm trong việc tạo máy ảo Android để kiểm thử.

- 1) Cài đặt Android Studio

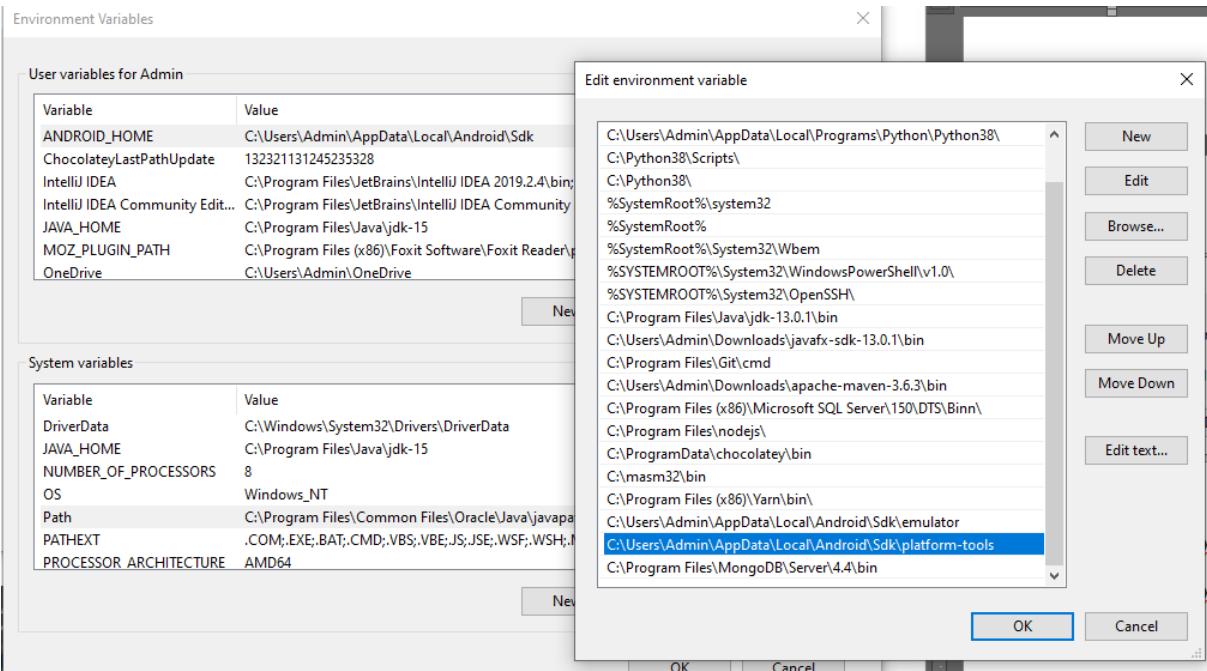


Hình 2.7. Cài đặt Android SDK qua Android Studio (1)

- 2) Chạy file thực thi và làm theo hướng dẫn cài đặt. Các packages cần lựa chọn để cài đặt: Android SDK Tools, Android SDK Platform-tools, Android SDK build-tools
- 3) Thêm biến môi trường



Hình 2.8. Cài đặt Android SDK (2)



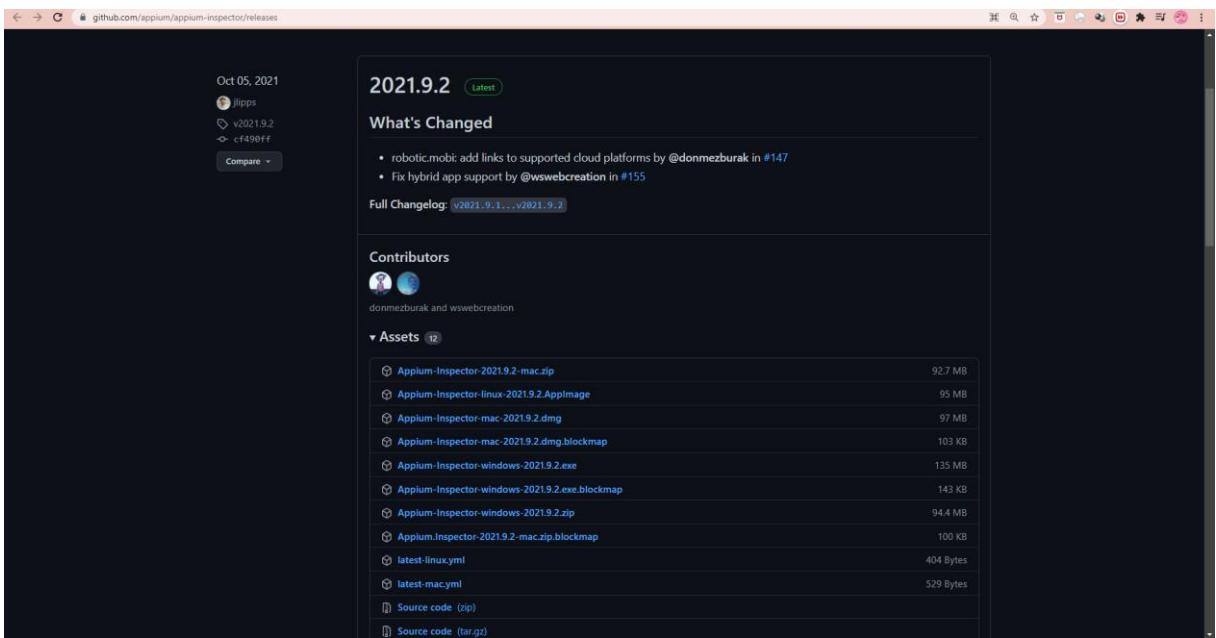
Hình 2.9. Cài đặt Android SDK (3)

2.4. Appium Inspector

Appium Inspector là một ứng dụng máy khách Appium với giao diện người dùng giúp chỉ định Appium Server kết nối tới, xác định các khả năng mong muốn (desired capabilities) sẽ được thiết lập, sau đó tương tác với các phần tử và các lệnh Appium khác sau khi bắt đầu một phiên. Appium Inspector chỉ hỗ trợ nền tảng Android và IOS, không hỗ trợ Windows

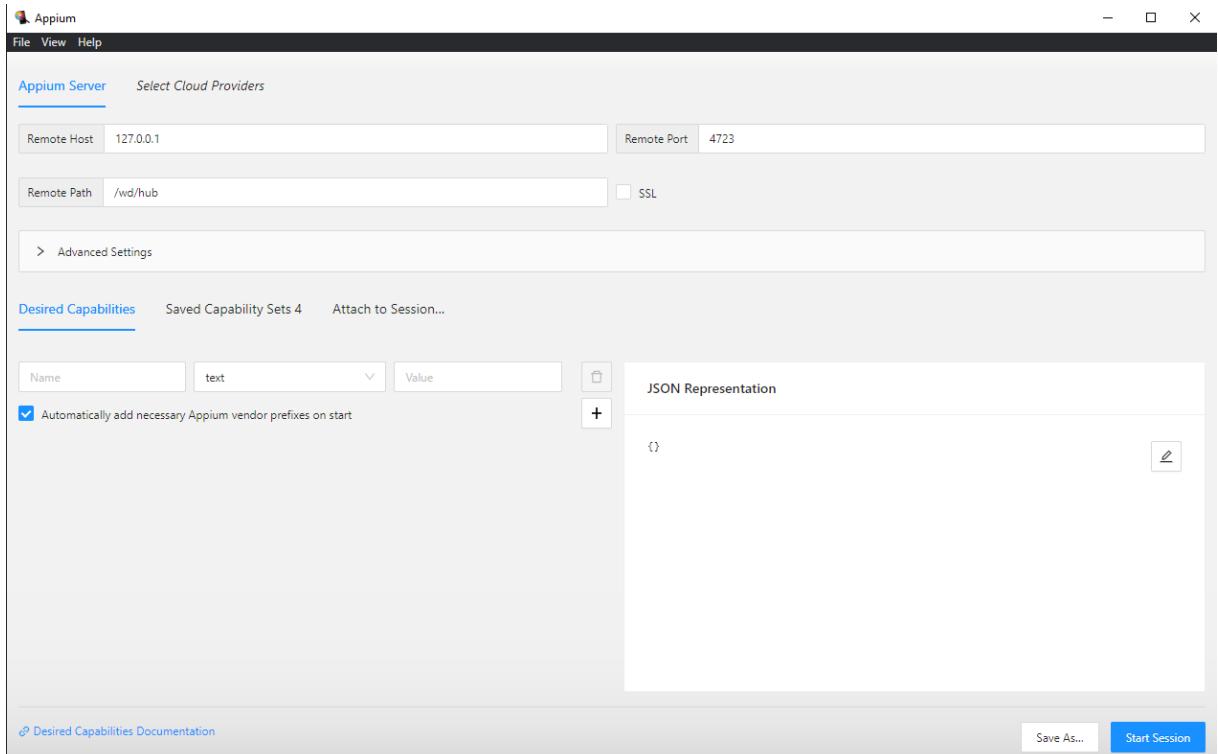
Cài đặt Appium Inspector

- 1) Truy cập <https://github.com/appium/appium-inspector/releases> và tải phiên bản Appium Inspector phù hợp với hệ điều hành.



Hình 2.10. Cài đặt Appium Inspector

2) Chạy file thực thi vừa tải để cài đặt.

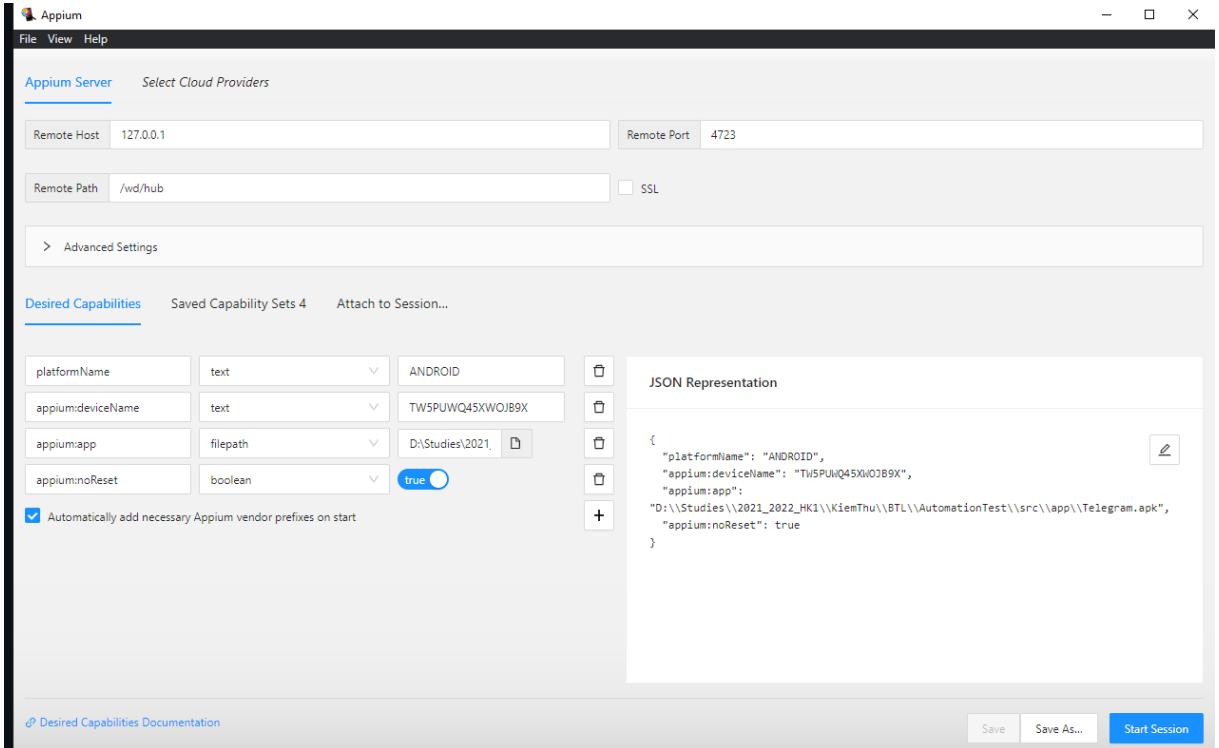


Hình 2.11. Giao diện Appium Inspector

Sử dụng Appium Inspector

- Nhập host và port của Appium Server , đường dẫn mặc định là /wd/hub
- Thiết lập các cặp giá trị
 - platformName: hệ điều hành IOS/Android

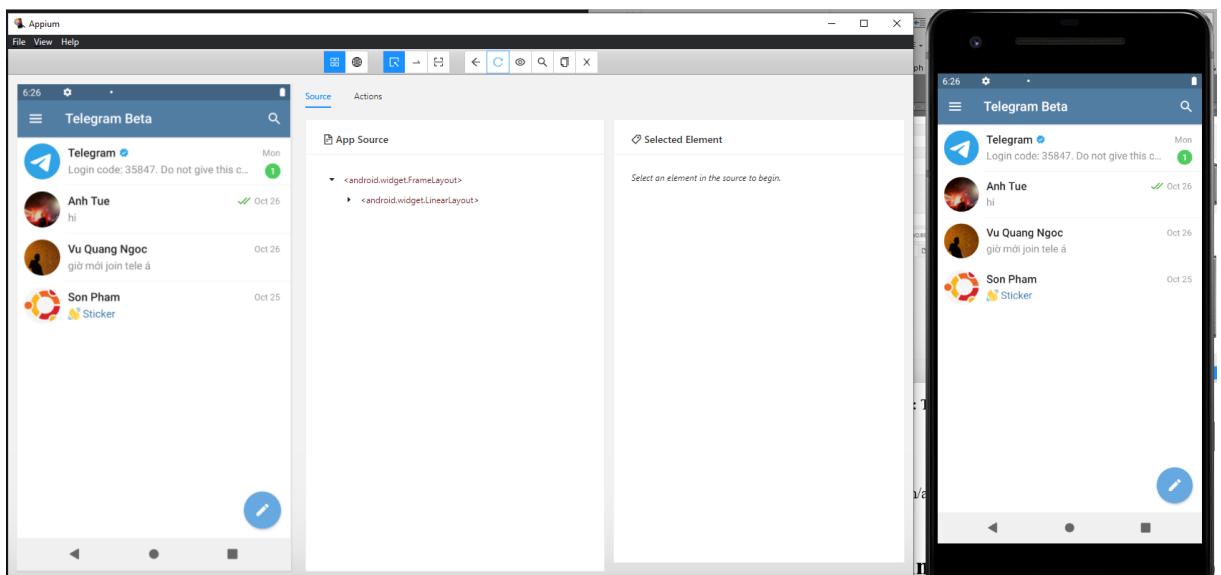
- deviceName: tên thiết bị kiểm thử (sử dụng lệnh adb devices để lấy được tên các thiết bị được kết nối)
- app: đường dẫn tới file apk của ứng dụng kiểm thử
- noReset: không khởi động lại ứng dụng trong mỗi phiên kiểm thử.



Hình 2.12. Appium Inspector: Thiết lập các giá trị cho Desired Capabilities

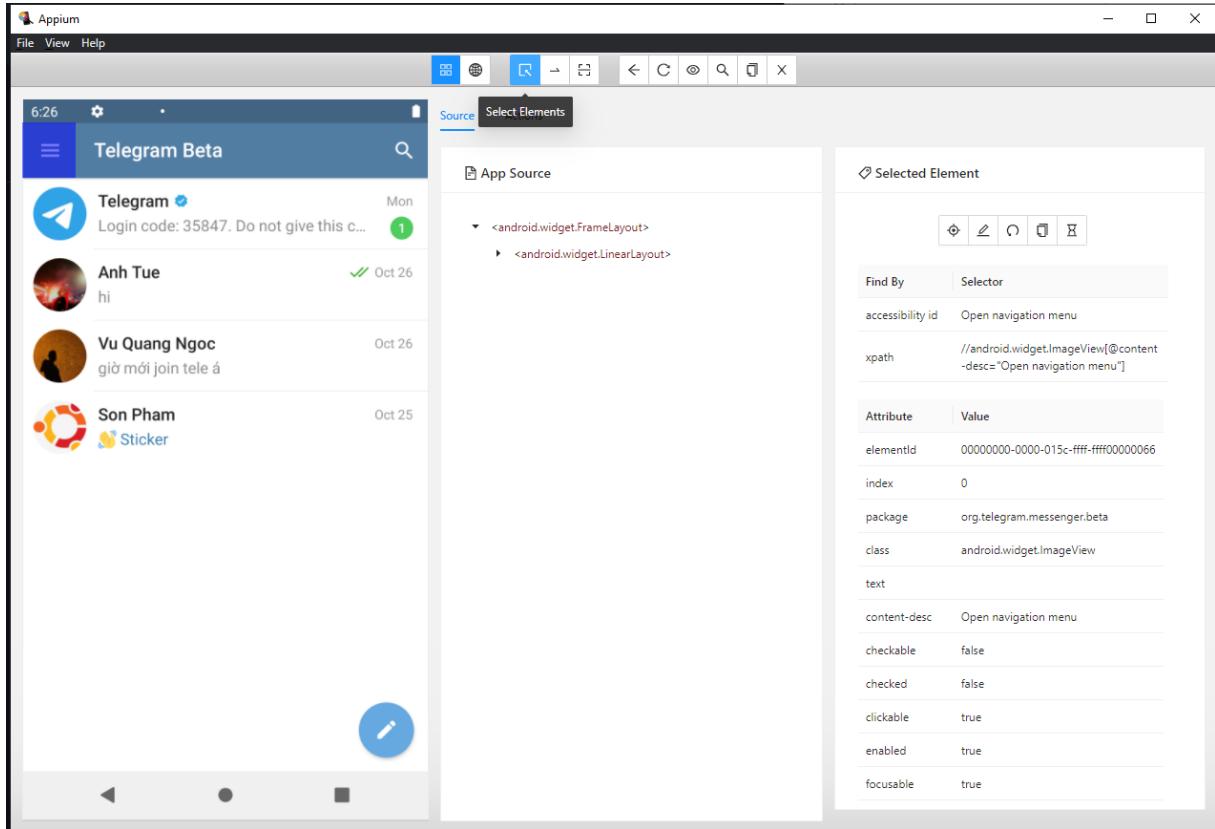
Bắt đầu một phiên, đảm bảo thiết bị di động kiểm thử được kết nối (ở đây nhóm dùng thiết bị Android ảo trên Android Studio) và Appium Server đã được bắt đầu.

Sau khi bắt đầu phiên, Appium Server thực hiện cài đặt file apk vào emulator. Ở đây nhóm đã đăng nhập sẵn vào ứng dụng.



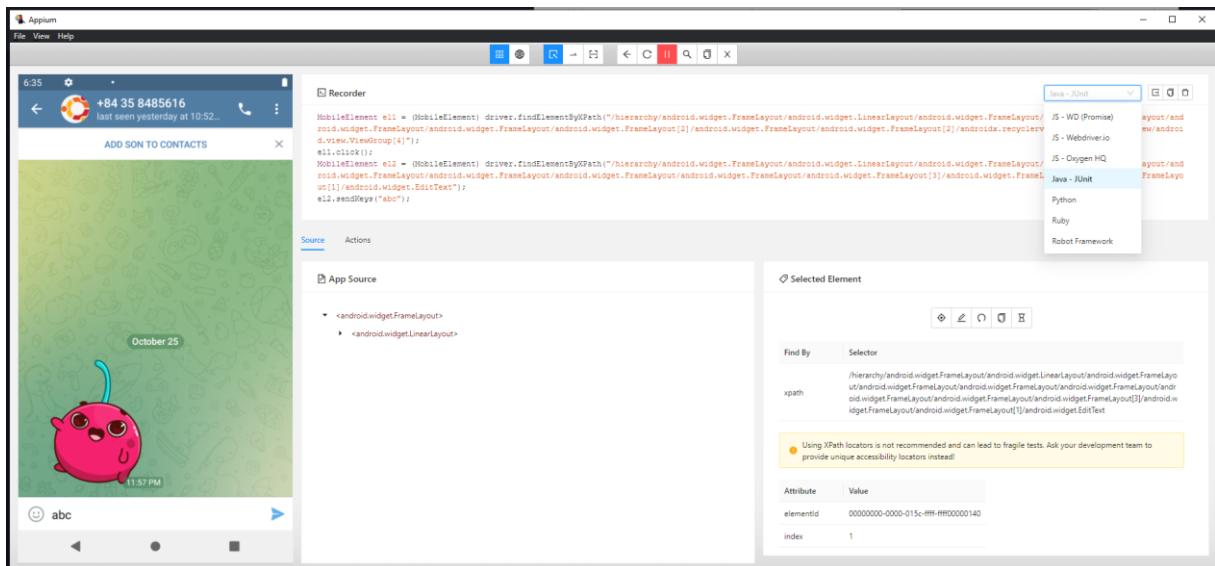
Hình 2.13. Appium Inspector: Bắt đầu phiên

Sử dụng chức năng Select Element để chọn các phần tử trên ứng dụng, từ đó, các cách định danh phần tử ứng dụng sẽ xuất hiện trên mục bên phải màn hình (Selected Element)



Hình 2.14. Appium Inspector: Chức năng Select Elements

Ngoài ra, chức năng Recording sẽ giúp ghi lại hoạt động của người dùng trên ứng dụng dưới dạng các câu lệnh với các ngôn ngữ tùy chọn như Java, Python, ...



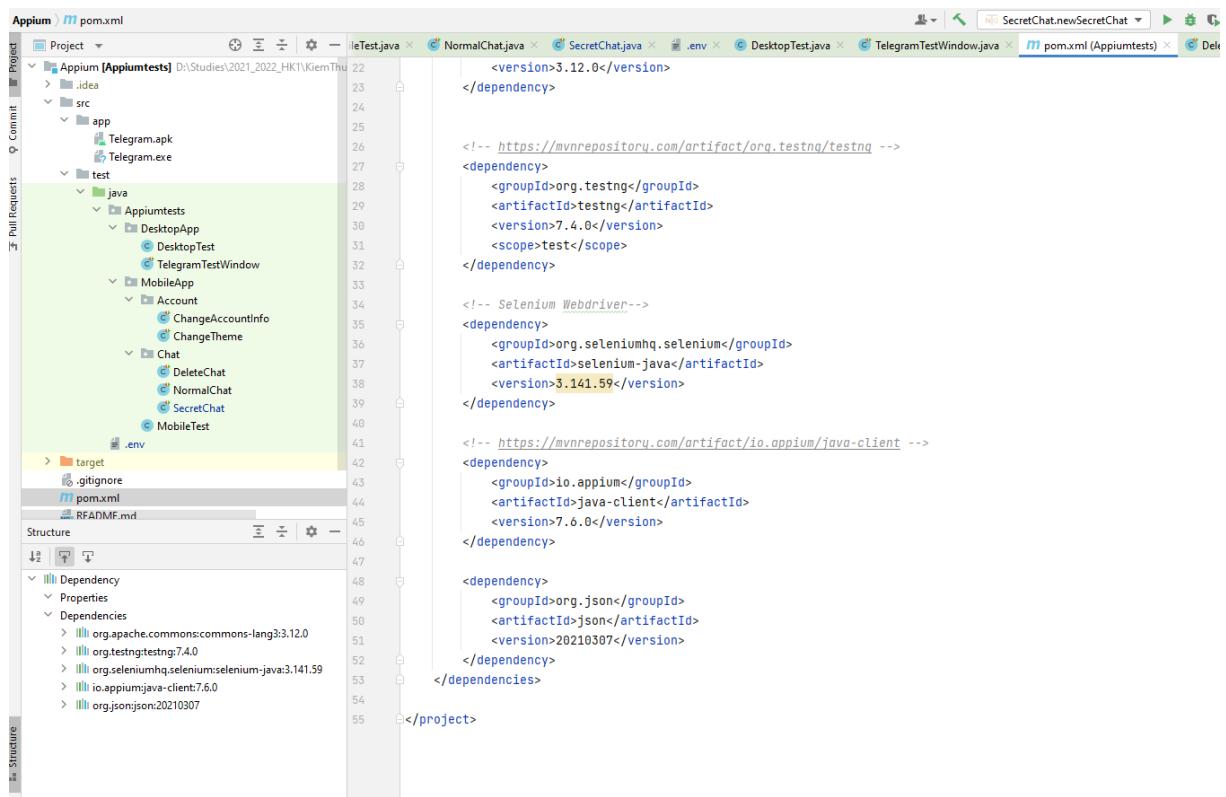
Hình 2.15. Appium Inspector: Chức năng Recording

2.5. Appium Client

Nhóm sử dụng ngôn ngữ Java để viết các tập lệnh kiểm thử, do đó, thư viện máy khách Appium được sử dụng là java-client.

Sử dụng IDE IntelliJ, nhóm tạo một Maven Project để viết các tập lệnh kiểm thử, thông qua việc thêm các dependency trong file pom.xml để cài đặt:

- TestNG
- Selenium-java: Selenium WebDriver hỗ trợ viết các API tự động bằng Java
- Java-client: thư viện máy khách hỗ trợ viết mã nguồn các ca kiểm thử bằng ngôn ngữ java



The screenshot shows the IntelliJ IDEA interface with the project 'Appium [Appiumtests]' open. The left sidebar shows the project structure with packages like 'src' and 'test'. The right pane displays the 'pom.xml' file content, which includes dependencies for TestNG, Selenium WebDriver, and the Appium Java Client. The code is color-coded for syntax highlighting.

```

<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.4.0</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
</dependency>

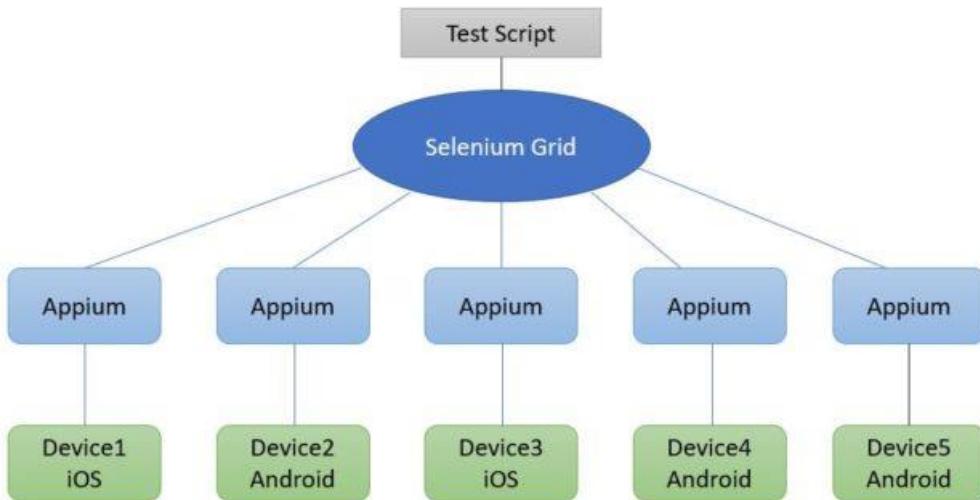
<dependency>
    <groupId>io.appium</groupId>
    <artifactId>java-client</artifactId>
    <version>7.6.0</version>
</dependency>

```

Hình 2.16. IntelliJ: Tạo Maven Project viết các tập lệnh sử dụng ngôn ngữ Java

2.6. Appium Grid

Appium Grid là sự kết hợp sử dụng Appium và Selenium Grid dùng để thực hiện các ca kiểm thử song song trên nhiều thiết bị khác nhau.



Hình 2.17. Kiến trúc Appium Grid

Cài đặt Selenium Grid

- Tải selenium grid bản [3.9.1](#) (bản 4.0 có một số thay đổi trong câu lệnh)
- Truy cập thư mục chứa tệp tin vừa tải, chạy câu lệnh dưới đây để tạo hub cho Selenium Grid với cấu hình mặc định, chạy ở <http://localhost:4444>

java -jar selenium-server-standalone-3.9.1.jar -role hub

- Truy cập <http://localhost:4444/grid/console> để kiểm tra, nếu Selenium Grid đã hoạt động sẽ hiển thị cấu hình tương tự như sau:

```

Grid Console v.3.9.1
Help

Config for the hub :
browserTimeout:0
debug:false
help:false
port:4444
role:hub
timeout:1800
cleanUpCycle:5000
host:192.168.1.9
capabilityMatcher:org.openqa.grid.internal.utils.DefaultCapabilityMatcher
newSessionWaitTimeout:-1
throwOnCapabilityNotPresent:true
registry:org.openqa.grid.internal.DefaultGridRegistry

Config details :
hub launched with : -browserTimeout 0 -debug false -help false -port 4444 -role hub -timeout 1800 -cleanUpCycle 5000 -host 192.168.1.9 -capabilityMatcher org.openqa.grid.internal.utils.DefaultCapabilityMatcher
-newSessionWaitTimeout -1 -throwOnCapabilityNotPresent true -registry org.openqa.grid.internal.DefaultGridRegistry
the final configuration comes from :
the default :
browserTimeout:0
debug:false
help:false
port:4444
role:hub
timeout:1800
cleanUpCycle:5000
capabilityMatcher:org.openqa.grid.internal.utils.DefaultCapabilityMatcher
newSessionWaitTimeout:-1
throwOnCapabilityNotPresent:true
registry:org.openqa.grid.internal.DefaultGridRegistry

updated with params :
browserTimeout:0
debug:false
help:false
port:4444
role:hub
timeout:1800
cleanUpCycle:5000
capabilityMatcher:org.openqa.grid.internal.utils.DefaultCapabilityMatcher
newSessionWaitTimeout:-1
throwOnCapabilityNotPresent:true
registry:org.openqa.grid.internal.DefaultGridRegistry

```

Hình 2.18. Selenium Grid Console

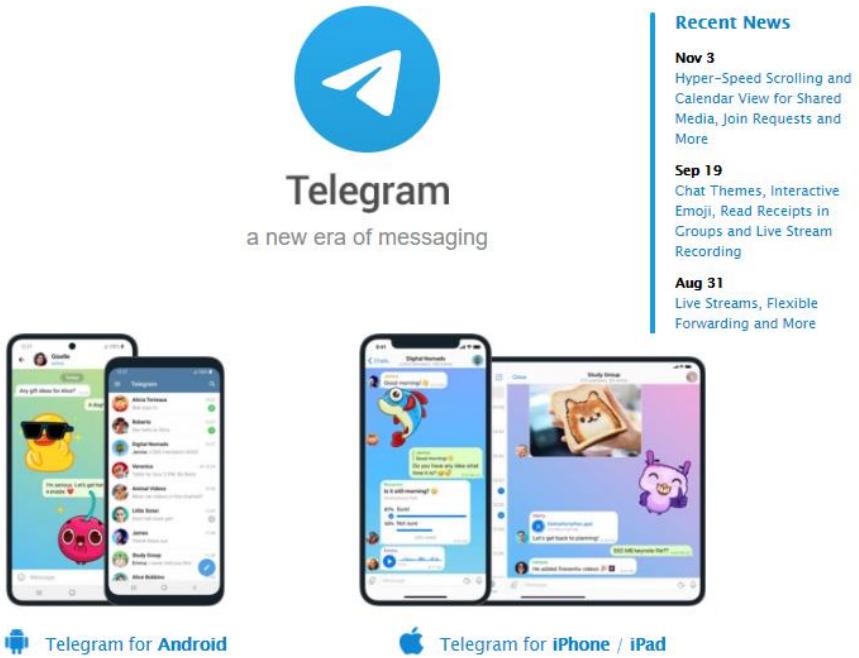
3. Kiểm thử ứng dụng

Trong phần này, nhóm sẽ giới thiệu một ứng dụng và các bước thực hiện kiểm thử tự động hóa sử dụng Appium trên ứng dụng đó.

3.1. Ứng dụng Telegram

Telegram là một ứng dụng nhắn tin, gọi điện, chia sẻ tệp đa nền tảng và miễn phí. Với Telegram, chúng ta có thể gửi tin nhắn, ảnh, video và tệp thuộc bất kỳ loại nào (doc, zip, mp3, ...) cũng như tạo nhóm trò chuyện với số lượng lên tới 200.000 người. Telegram có thể đáp ứng tất cả nhu cầu nhắn tin cá nhân hay công việc, đồng thời hỗ trợ cuộc gọi thoại và video mã hóa đầu cuối.

Nhóm lựa chọn Telegram bởi nó có đầy đủ các phiên bản ứng dụng chạy trên các thiết bị IOS/Android, Window Desktop và cả phiên bản web app. Điều này phù hợp để kiểm thử đa nền tảng với Appium.



Telegram for PC / Linux

Telegram for macOS

Ngoài ra, với giao diện đơn giản, thân thiện và dễ sử dụng, Telegram sẽ giúp việc demo kiểm thử của nhóm dễ dàng hơn.

3.2. Chức năng

Để tiến hành viết các ca kiểm thử, nhóm xác định các tính năng nổi bật của Telegram, từ đó lựa chọn một số tính năng phù hợp, có thể dễ dàng viết các ca kiểm thử. Và các tính năng được nhóm lựa chọn để kiểm thử gồm:

- Tìm kiếm bạn bè, truy cập cuộc trò chuyện
- Các tác vụ nhắn tin thông thường, gồm có:
 - Gửi tin nhắn văn bản
 - Gửi biểu tượng cảm xúc
 - Gửi nhãn dán
 - Gửi ảnh

- Các tác vụ bổ sung:
 - Chia sẻ vị trí hiện tại trong cuộc trò chuyện
 - Thay đổi giao diện ứng dụng
- Cập nhật thông tin cá nhân
- Tạo cuộc trò chuyện bí mật
- Xóa cuộc trò chuyện

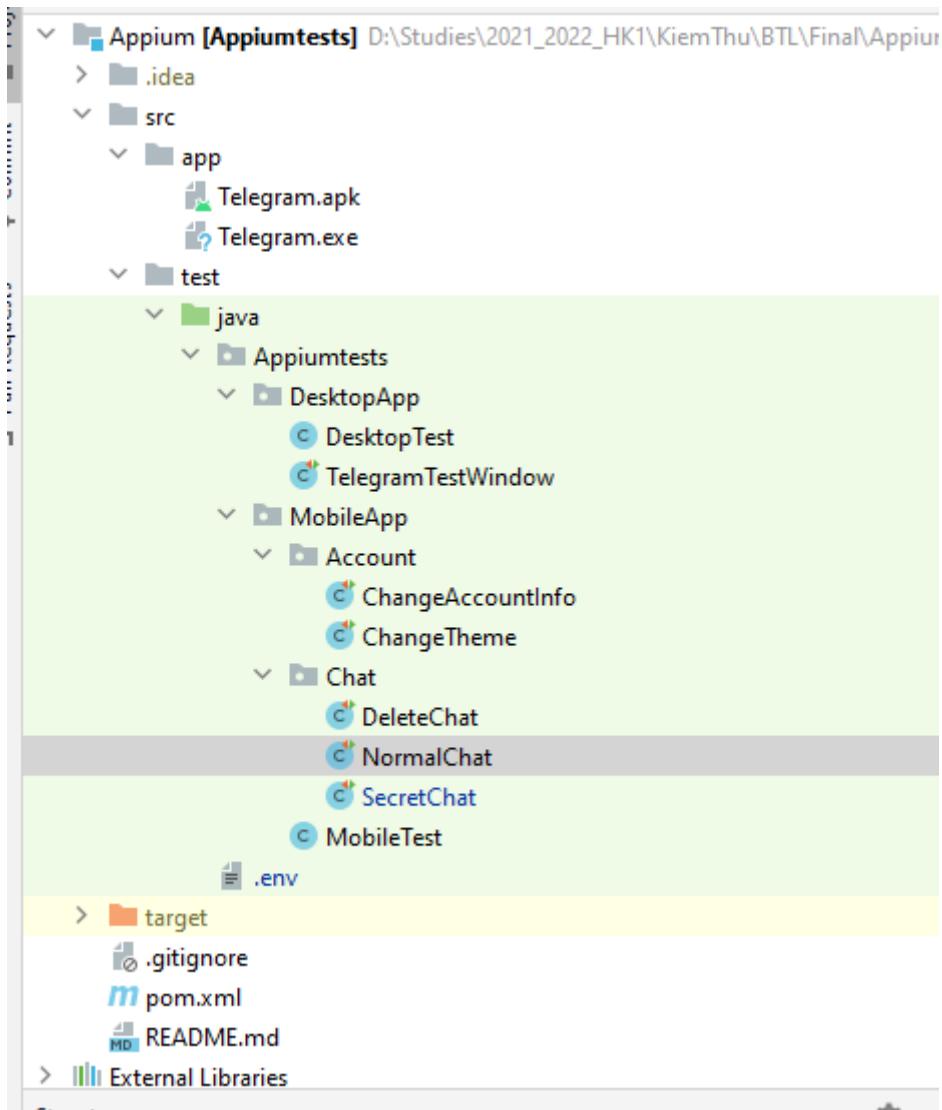
3.3. Các ca kiểm thử

Như đã đề cập trong phần cài đặt, nhóm sử dụng ngôn ngữ Java và IDE IntelliJ để viết các ca kiểm thử.

Cấu trúc mã nguồn được chia thành hai package MobileApp để kiểm thử ứng dụng trên thiết bị di động Android/IOS và package DesktopApp để demo thực hiện kiểm thử trên Windows. Bên cạnh đó, package ứng dụng Telegram được lưu tại //src//app// với file Telegram.apk được dùng để cài trên thiết bị Android, file Telegram.exe được dùng để cài trên thiết bị máy tính Windows. Các file này đều được tải xuống từ <https://telegram.org/apps>

Mã nguồn tập lệnh kiểm thử được nhóm viết tại

<https://github.com/huyenlovemath/Automation-Testing>



Hình 3.1. Cấu trúc mã nguồn của các tập lệnh kiểm thử

Package Appiumtests:

➤ **MobileApp**

- **MobileTest.java**: thiết lập các desired capabilities và kết nối tới Appium Server tạo phiên kiểm thử.

➤ **Account**

- **ChangeAccountInfo.java**: với các testcase thay đổi thông tin người dùng gồm họ tên và tiểu sử.
- **ChangeTheme.java**: với các testcase thay đổi chủ đề giao diện người dùng: chuyển đổi chế độ ban ngày sang ban đêm hoặc ngược lại

➤ **Chat**

- **NormalChat.java**: với các testcase gửi tin nhắn (text, emoji, sticker, image); chia sẻ vị trí hiện tại, chỉnh sửa ảnh trước khi gửi.

- **SecretChat.java:** với testcase tạo cuộc trò chuyện bí mật
- **DeleteChat.java:** với các testcase xóa một hoặc nhiều cuộc trò chuyện.

➤ DesktopApp

- **DesktopTest.java:** thiết lập các desired capabilities và kết nối tới Appium Server tạo phiên kiểm thử.
- **TelegramTestWindow.java:** với testcase thay đổi giao diện ứng dụng

3.3.1. Các ca kiểm thử cho chức năng gửi tin nhắn

1) Tìm kiếm user và truy cập tin nhắn

Bước 1: Nhập vào biểu tượng kính lúp để tìm kiếm

Bước 2: Nhập vào vùng nhập để bắt đầu nhập

Bước 3: Nhập tên user bất kỳ

Bước 4: Nhập vào user đầu tiên tìm thấy

```
@Test
public void FindUserAndCheckInbox() {
    System.out.println(" Start Find User test");

    WebElement searchButton = driver.findElementByXPath( using: "//android.widget.ImageButton[@content-desc='Search']");
    searchButton.click();
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

    WebElement edit = driver.findElementByXPath( using: "//android.widget.EditText[@text='Search']");
    edit.click();
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

    edit.sendKeys( ...charSequences: "Son");
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

    WebElement chooseUser = driver.findElementByXPath( using: "//android.view.ViewGroup[@index='0']");
    chooseUser.click();
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

}
```

Hình 3.2. MobileTest: Tìm kiếm người dùng và truy cập tin nhắn

2) Nhắn tin: Văn bản

Bước 1: Lựa chọn người đầu tiên trong danh sách để nhắn tin

Bước 2: Nhập vào vùng nhập để bắt đầu nhập

Bước 3: Nhập một đoạn văn bản bất kỳ

Bước 4: Nhập vào button Send để gửi tin nhắn đi

Bước 5: So sánh kết quả

```

@Test
public void SendMessage() {
    System.out.println(" Start Send Message test");

    // sending msg

    String messageToSend = "abc";
    WebElement choosePartner = driver.findElementByXPath( using: "//android.view.ViewGroup[@index='0']");
    choosePartner.click();
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
    WebElement edit = driver.findElementByXPath( using: "//android.widget.EditText[@index='1']");
    edit.click();
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
    edit.sendKeys(messageToSend);
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
    WebElement sendButton = driver.findElementByXPath( using: "//android.view.View[@content-desc='Send']");
    sendButton.click();

    driver.manage().timeouts().implicitlyWait( 10, TimeUnit.SECONDS);

    // checking msg

    List<WebElement> message = driver.findElementsByXPath( using: "//androidx.recyclerview.widget.RecyclerView/*");
    WebElement lastMessage = message.get(message.size() - 1);
    String lastMessageString = lastMessage.getAttribute ( s: "content-desc").toString();
    Assert.assertEquals(lastMessageString.substring(0, messageToSend.length()),messageToSend);
}

```

Hình 3.3. MobileTest: Nhắn tin văn bản

3) Nhắn tin: Gửi biểu tượng cảm xúc

Bước 1: Lựa chọn người đầu tiên trong danh sách để nhắn tin

Bước 2: Nhập vào button dưới cùng bên trái để lựa chọn gửi biểu tượng cảm xúc

Bước 3: Lựa chọn “Gửi biểu cảm”

Bước 4: Chọn 1 biểu cảm bất kỳ

Bước 5: Nhập vào button Send để gửi tin nhắn đi

Bước 6: So sánh kết quả

```

    @Test
    public void SendEmoji() {
        System.out.println(" Start Send Message test");

        String emojiToSend = "\uD83D\uDE18";
        WebElement choosePartner = driver.findElementByXPath( using: "//android.view.ViewGroup[@index='0']");
        choosePartner.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
        WebElement emojiButton = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='Emoji, stickers, and GIFs']");
        emojiButton.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
        WebElement emojisButton = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='Emoji']");
        emojisButton.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
        WebElement emoji = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='\uD83D\uDE18']");
        emoji.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
        WebElement sendButton = driver.findElementByXPath( using: "//android.view.View[@content-desc='Send']");
        sendButton.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

        driver.navigate().back();
        List<WebElement> message = driver.findElementsByXPath( using: "//androidx.recyclerview.widget.RecyclerView/*");
        WebElement lastMessage = message.get(message.size() - 1);
        String lastMessageString = lastMessage.getAttribute ( s: "content-desc").toString();
        Assert.assertEquals(lastMessageString.substring(0, emojiToSend.length()), emojiToSend);
    }
}

```

Hình 3.4. MobileTest: Nhắn biểu tượng cảm xúc

4) Nhắn tin: Gửi nhãn dán

Bước 1: Lựa chọn người muốn nhắn tin

Bước 2: Nhập vào button dưới cùng bên trái để lựa chọn gửi nhãn dán

Bước 3: Lựa chọn “Gửi nhãn dán”

Bước 4: Chọn 1 nhãn dán bất kỳ

Bước 5: Nhập vào button Send để gửi tin nhắn đi

Bước 6: So sánh kết quả

```

    @Test
    public void SendStiker() {
        System.out.println(" Start Send Message test");

        String stickerToSend = "\uD83D\uDE02 Sticker";
        WebElement choosePartner = driver.findElementByXPath( using: "//android.view.ViewGroup[@index='0']");
        choosePartner.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
        WebElement emojiButton = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='Emoji, stickers, and GIFs']");
        emojiButton.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
        WebElement stickerButton = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='Stickers']");
        stickerButton.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
        WebElement stiker = driver.findElementByXPath( using: "//android.widget.FrameLayout[@content-desc='\uD83D\uDE02 Sticker']");
        stiker.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

        driver.navigate().back();
        List<WebElement> message = driver.findElementsByXPath( using: "//androidx.recyclerview.widget.RecyclerView/*");
        WebElement lastMessage = message.get(message.size() - 1);
        String lastMessageString = lastMessage.getAttribute ( s: "content-desc").toString();
        Assert.assertEquals(lastMessageString.substring(0, stickerToSend.length()), stickerToSend);
    }
}

```

Hình 3.5. MobileTest: Gửi nhãn dán

5) Nhắn tin: Gửi hình ảnh

Bước 1: Lựa chọn người muốn gửi ảnh

Bước 2: Nhấp vào button đính kèm

Bước 3: Chọn ảnh muốn gửi

Bước 4: Nhấp button Send để gửi ảnh

```
@Test
public void SendImage() {
    System.out.println(" Start Send Image test");

    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

    WebElement choosePartner = driver.findElementByXPath( using: "//android.view.ViewGroup[@index='0']");
    choosePartner.click();
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
    WebElement chooseImage = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='Attach media']");
    chooseImage.click();
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

    //get images
    WebElement sendImage = (WebElement) driver.findElementsByXPath( using: "/hierarchy/android.widget.FrameLayout/android.widg
    sendImage.click();
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

    WebElement sendButton = (WebElement) driver.findElementById( using: "Send");
    sendButton.click();
    sendButton.click();
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

}
```

Hình 3.6. MobileTest: Gửi hình ảnh

6) Chính sửa ảnh trước khi gửi

Bước 1: Lựa chọn người muốn gửi ảnh

Bước 2: Nhấp vào button đính kèm

Bước 3: Chọn ảnh muốn gửi

Bước 4: Nhấp button Crop ảnh

Bước 5: Crop ảnh tùy ý (Trong test này xoay ảnh đi 90 độ)

Bước 6: Nhấp “Crop”

Bước 7: Nhấp button Send để gửi ảnh

```

    @Test
    public void CropBeforeSendImage() {
        System.out.println(" Start Crop Before Send Image test");

        WebElement choosePartner = driver.findElementByXPath( using: "//android.view.ViewGroup[@index='0']");
        choosePartner.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
        WebElement chooseImage = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='Attach media']");
        chooseImage.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

        WebElement sendImage = driver.findElementByXPath( using: "/hierarchy/android.widget.FrameLayout/android.widget.FrameLayout/andro:
sendImage.click();
driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

WebElement cropButton = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='Crop image']");
cropButton.click();
driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

WebElement rotate = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='Rotate']");
rotate.click();
driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

WebElement crop = driver.findElementByXPath( using: "//android.widget.TextView[@text='CROP']");
crop.click();
driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

WebElement sendButton = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='Send']");
sendButton.click();
driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

}

```

Hình 3.7. MobileTest: Chính sửa ảnh trước khi gửi

7) Chia sẻ địa chỉ

Bước 1: Lựa chọn người bạn muốn chia sẻ địa chỉ

Bước 2: Click vào button đính kèm

Bước 3: Chọn mục location

Bước 4: Chọn “Send my current location”

```

    @Test
    public void ShareLocationMessage() {
        System.out.println(" Start Share Location test");

        String locationString ="Location";
        WebElement choosePartner = driver.findElementByXPath( using: "//android.view.ViewGroup[@index='0']");
        choosePartner.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
        WebElement chooseLocation = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='Attach media']");
        chooseLocation.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

        WebElement sendLocation = driver.findElementByXPath( using: "//android.widget.TextView[@text='Location']");
        sendLocation.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

        WebElement SEND = driver.findElementByXPath( using: "//android.widget.TextView[@text='Send My Current Location']");
        SEND.click();
        driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

        //List<WebElement> message = driver.findElementsByXPath("//androidx.recyclerview.widget.RecyclerView/*");
        //WebElement lastMessage = message.get(message.size() - 1);
        //String lastMessageString = lastMessage.getAttribute ("content-desc").toString();
        //String lastMessageString = message.get(0).getAttribute ("index").toString();

        //System.out.println(lastMessageString);
        //Assert.assertEquals(lastMessageString.substring(0, locationString.length()),locationString);
    }
}

```

Hình 3.8. MobileTest: Chia sẻ địa chỉ

8) Xóa một cuộc trò chuyện:

Bước 1: Lựa chọn cuộc trò chuyện muốn xóa

Bước 2: Chọn và nhấn giữ vào cuộc trò chuyện, đợi cho đến khi xuất hiện dấu tích xanh

Bước 3: Nhấp vào biểu tượng thùng rác để xóa cuộc trò chuyện

Bước 4: Nhấp chọn DELETE CHAT trên cửa sổ thông báo để xác nhận xóa

Bước 5: So sánh số cuộc trò chuyện trước và sau khi thực hiện xóa.

```

    @Test
    public void deleteOneChat() {
        int totalChats = this.driver.findElementsByClassName( using: "android.view.ViewGroup").size();
        System.out.println("Total chat " + totalChats);
        WebElement chatForDelete = this.driver.findElementByXPath( using: "//android.view.ViewGroup[1]");

        Actions actions = new Actions(this.driver);
        actions.clickAndHold(chatForDelete).perform();

        WebElement buttonMoveToTrash = this.driver.findElementByXPath( using: "//android.widget.ImageButton[@content-desc='Delete']/android.widget.ImageView");
        buttonMoveToTrash.click();
        this.driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

        WebElement deleteChat = this.driver.findElementByXPath( using: "//android.widget.TextView[@text='DELETE CHAT']");
        deleteChat.click();
        this.driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

        int totalChatsAfterDelete = this.driver.findElementsByClassName( using: "android.view.ViewGroup").size();
        System.out.println("Total chat after delete " + totalChatsAfterDelete);

        Assert.assertEquals(totalChatsAfterDelete, expected: totalChats-1);
    }
}

```

Hình 3.9. MobileTest: Xóa một cuộc trò chuyện

9) Xóa nhiều cuộc trò chuyện

Bước 1: Lựa chọn các cuộc trò chuyện muốn xóa

Bước 2: Chọn và nhấn giữ chuột vào cuộc trò chuyện đầu tiên muốn xóa, đợi cho đến khi xuất hiện dấu tích xanh

Bước 3: Nhấp vào các cuộc trò chuyện tiếp theo muốn xóa

Bước 4: Nhấp chọn biểu tượng thùng rác để xóa cuộc trò chuyện

Bước 5: Nhấp chọn DELETE trên cửa sổ để xác nhận xóa các cuộc trò chuyện

Bước 5: So sánh số cuộc trò chuyện trước và sau khi xóa

```
@Test
public void deleteMultiChats() {
    int totalChats = this.driver.findElementsByClassName(using: "android.view.ViewGroup").size();
    System.out.println("Total chat " + totalChats);
    WebElement chatForDelete1 = this.driver.findElementByXPath(using: "//android.view.ViewGroup[1]");
    WebElement chatForDelete2 = this.driver.findElementByXPath(using: "//android.view.ViewGroup[2]");

    Actions actions = new Actions(this.driver);
    actions.clickAndHold(chatForDelete1).perform();
    chatForDelete2.click();
    this.driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);

    WebElement buttonMoveToTrash = this.driver.findElementByXPath(using: "//android.widget.ImageButton[@content-desc=\"Delete\"]");
    buttonMoveToTrash.click();
    this.driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);

    WebElement deleteChat = this.driver.findElementByXPath(using: "//android.widget.TextView[@text=\"DELETE\"]");
    deleteChat.click();
    this.driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);

    int totalChatsAfterDelete = this.driver.findElementsByClassName(using: "android.view.ViewGroup").size();
    System.out.println("Total chat after delete " + totalChatsAfterDelete);

    Assert.assertEquals(totalChatsAfterDelete, expected: totalChats-2);
}
```

Hình 3.10. MobileTest: Xóa nhiều cuộc trò chuyện

3.3.2. Các ca kiểm thử cho chức năng cài đặt thông tin, giao diện người dùng

1) Thay đổi giao diện người dùng

Bước 1: Click button Menu

Bước 2: Click button thay đổi theme ứng dụng sang chế độ sáng hoặc tối.

```

@Test
public void ChangeTheme() {
    System.out.println(" Start Change Theme test");

    WebElement Menu = driver.findElementByXPath( using: "//android.widget.ImageView[@content-desc='Open navigation menu']");
    Menu.click();
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);
    WebElement changeTheme = driver.findElementByXPath( using: "//android.widget.ImageView[@text='Switch to day theme']");
    changeTheme.click();
    driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

}

```

Hình 3.11. MobileTest: Thay đổi giao diện

2) Thay đổi thông tin cá nhân người dùng:

Trước khi thực hiện ca kiểm thử, tiến hành mở mục quản lý thông tin tài khoản trước:

```

public class ChangeAccountInfo extends MobileTest {

    public ChangeAccountInfo() throws IOException {
    }

    private static final String firstNameTest = "NewFirstName";
    private static final String lastNameTest = "NewLastName";
    private static final String bioTest = "New Bio";
    private static final int maxLengthBio = 70;

    @BeforeMethod
    private void openAccountInfo() {
        System.out.println("Open account details");
        WebElement navigationMenu = this.driver.findElementByAccessibilityId( using: "Open navigation menu");
        navigationMenu.click();
        this.driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

        WebElement buttonAccountInfo = this.driver.findElementByXPath( using: "//android.widget.FrameLayout[1]/android.view.View");
        buttonAccountInfo.click();
        this.driver.manage().timeouts().implicitlyWait( 3, TimeUnit.SECONDS);

        this.driver.findElementByXPath( using: "//android.widget.TextView[@text='Account']");
        this.driver.findElementByXPath( using: "//android.widget.TextView[@text='Settings']");
    }
}

```

Hình 3.12. BeforeMethod: Mở mục quản lý thông tin tài khoản

Thực hiện thay đổi họ tên tài khoản:

Bước 1: Nhập vào biểu tượng Menu ở góc trái màn hình

Bước 2: Nhập vào hình đại diện tài khoản

Bước 3: Nhập nhọn biểu tượng More Options ở góc phải trên màn hình

Bước 4: Nhập chọn Edit name

Bước 5: Điền First name và Last name vào ô tương ứng

Bước 6: Chọn dấu tích để xác nhận thay đổi

```

    @Test
    public void EditName() {
        System.out.println("Test edit name");

        WebElement buttonMoreInfo = this.driver.findElementByXPath(using: "//android.widget.ImageButton[@content-desc=\"More options\"]/android.widget.ImageView");
        buttonMoreInfo.click();
        this.driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);

        WebElement buttonEditName = this.driver.findElementByXPath(using: "//android.widget.TextView[@text=\"Edit name\"]");
        buttonEditName.click();
        this.driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);

        WebElement firstName = this.driver.findElementByXPath(using: "//android.widget.EditText[1]");
        WebElement lastName = this.driver.findElementByXPath(using: "//android.widget.EditText[2]");
        firstName.clear();
        lastName.clear();
        firstName.sendKeys(firstNameTest);
        lastName.sendKeys(lastNameTest);

        WebElement saveName = this.driver.findElementByXPath(using: "//android.widget.ImageButton[@content-desc=\"Done\"]");
        saveName.click();
        this.driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);

        WebElement fullName = this.driver.findElementByXPath(using: "//android.widget.FrameLayout[1]/android.widget.TextView[1]");
        fullName.getText();
        Assert.assertEquals(fullName.getText(), expected: firstNameTest + " " + lastNameTest);
    }
}

```

Hình 3.13. MobileTest: Thay đổi tên người dùng

Thực hiện thay đổi tiêu sử (mô tả) cá nhân:

Bước 1: Nhập vào biểu tượng Menu ở góc trái màn hình

Bước 2: Nhập vào hình đại diện tài khoản

Bước 3: Nhập chọn mục Bio ở phần Account

Bước 4: Điền tiêu sử

Bước 5: Chọn dấu tích để xác nhận thay đổi

```

    @Test
    public void EditBio() {
        System.out.println("Test edit bio");

        WebElement bioTitle = this.driver.findElementByXPath(using: "//android.widget.TextView[@text=\"Bio\"]");
        WebElement bioContent = this.driver.findElementByXPath(using: "//android.widget.FrameLayout[5]");
        bioContent.click();
        this.driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);

        WebElement bioEditText = this.driver.findElementByXPath(using: "//android.widget.EditText");
        WebElement saveBioContent = this.driver.findElementByXPath(using: "//android.widget.ImageButton[@content-desc=\"Done\"]");
        if (bioTest.length() < maxLengthBio) {
            bioEditText.clear();
            bioEditText.sendKeys(bioTest);
        } else {
            Assert.fail("Bio content test is too long");
        }
        saveBioContent.click();
        this.driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);

        WebElement bioContentNew = this.driver.findElementByXPath(using: "//android.widget.FrameLayout[5]");
        this.driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);

        Assert.assertEquals(bioContentNew.getText(), expected: "Bio: " + bioTest);
    }
}

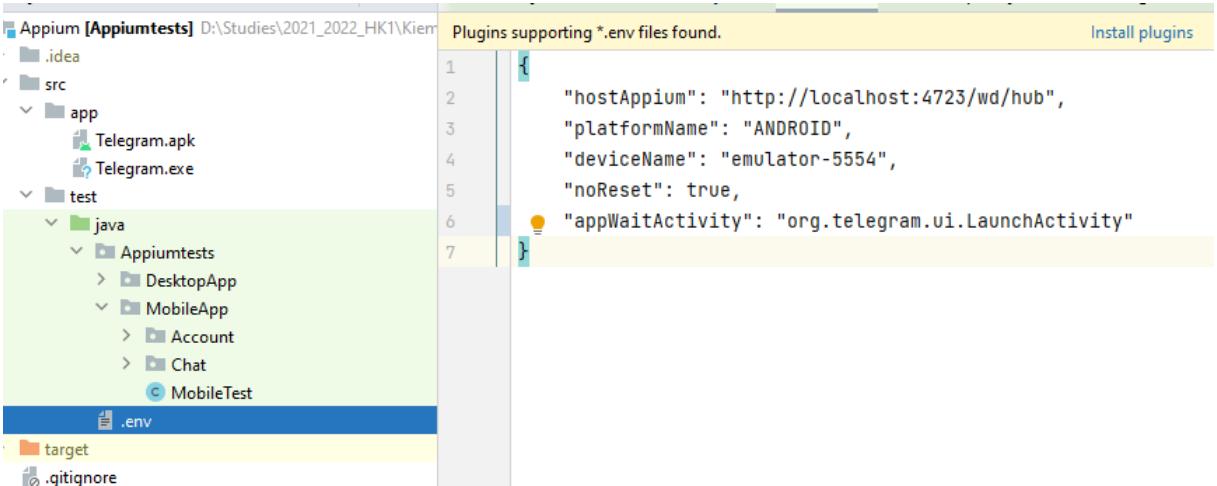
```

Hình 3.14. MobileTest: Thay đổi tiêu sử người dùng

3.3.3. Thực thi các ca kiểm thử trên nền tảng Android

Với các tập lệnh kiểm thử trong package MobileApp, có thể kiểm thử trên cả nền tảng IOS và Android.

File môi trường Appiumtests//MobileApp//.env được tạo ra để lưu các desired capabilities giúp thuận tiện khi thay đổi kiểm thử trên các thiết bị hoặc các nền tảng di động khác nhau (thay vì sửa đổi trực tiếp vào mã nguồn, chỉ cần chỉnh sửa trong file này là có thể thực hiện kiểm thử được).



The screenshot shows a code editor with an open file named '.env'. The file contains the following JSON configuration:

```
hostAppium": "http://localhost:4723/wd/hub",
platformName": "ANDROID",
deviceName": "emulator-5554",
noReset": true,
appWaitActivity": "org.telegram.ui.LaunchActivity"
```

The code editor interface includes a sidebar showing project structure with folders like .idea, src, app, test, and a .env file highlighted. A status bar at the top says 'Plugins supporting *.env files found.' and has an 'Install plugins' button.

Hình 3.15. Thiết lập để kiểm thử trên Android Device

Vì điều kiện vật chất của nhóm có hạn, nhóm chỉ tiến hành thực hiện kiểm thử trên thiết bị Android.

3.3.4. Thực thi ca kiểm thử trên nền tảng Windows:

Nhóm thực hiện viết một ca kiểm thử thay đổi giao diện người dùng trên ứng dụng Telegram cho Windows.

```

public class DesktopTest {
    private static String localDir = System.getProperty("user.dir");
    private static final String APP = localDir + "\\src\\app\\Telegram.exe";
    protected AndroidDriver driver;
    private static final String APPIUM = "http://localhost:4723/wd/hub";

    public DesktopTest() throws IOException {
    }

    @BeforeMethod
    public void SetUp() throws Exception {

        DesiredCapabilities caps = new DesiredCapabilities();
        caps.setCapability(capabilityName: "platformName", value: "Windows");
        caps.setCapability(capabilityName: "platformVersion", value: "10");
        caps.setCapability(capabilityName: "deviceName", value: "WindowsPC");
        caps.setCapability(capabilityName: "app", APP);

        try {
            driver = new AndroidDriver(new URL(APPIUM), caps);
        } catch (WebDriverException e) {
            System.out.println("Cannot get your device");
            e.printStackTrace();
        }
    }

    @AfterMethod
    public void TearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}

```

Hình 3.16. Kiểm thử trên Window Desktop (1)

Trước tiên tạo class DesktopTest chức năng tương tự class MobileTest, thực hiện xác định Desired Capibilities và kết nối Appium Server bắt đầu phiên kiểm thử. Ở đây, platformName, deviceName được thay đổi tương ứng với nền tảng Windows và thiết bị máy tính dùng cho kiểm thử ứng dụng.

Sau bước chuẩn bị trên, tạo class TelegramTestWindow mở rộng class DesktopTest, thực hiện ca kiểm thử Thay đổi giao diện người dùng:

```

public class TelegramTestWindow extends DesktopTest {

    private AppiumDriver<WebElement> driver;

    public TelegramTestWindow() throws IOException {
    }

    @Test
    public void TestChangeTheme() {

        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
        WebElement Menu = driver.findElementByXPath(using: "/Window/Group[2]/Group/Group[2]/Group[1]/Group");
        Menu.click();

        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
        WebElement ChangeTheme = driver.findElementByXPath(using: "/Window/Group[2]/Group/Group[2]/Group[2]/Group[4]/Group/Group/Group[3]/Group");
        ChangeTheme.click();

        driver.manage().timeouts().implicitlyWait(3, TimeUnit.SECONDS);
    }
}

```

Hình 3.17. Kiểm thử trên Window Desktop (2): Thay đổi giao diện người dùng

3.4. Kết quả kiểm thử:

Thực thi các ca kiểm thử trên thiết bị Android, kết quả trả về thành công trên các testcases trong thời gian 3m57s.

Appiumtests.MobileApp: 44 total, 44 passed		3 m 57 s
com.intelij.rt.testing RemoteTestNGStarter -usedefaultlisteners false -socketfd3500 @w@C:\Users\DELL\AppData\Local\Temp\idea_working_dirs_testing\tmp C:\Users\DELL\AppData\Local\Temp\idea_testing\tmp		
Suite tests run: 14, Passes: 14, Failures: 0, Skips: 0		Process finished with exit code 0
Automation-Testing-main		Collapse Expand
DeleteChat		3 m 57 s
DeleteChat.deleteMultiChats	passed	5.12 s
Total chat 3		
Total chat after delete 1		
DeleteChat.deleteOneChat	passed	4.76 s
Total chat 3		
Total chat after delete 2		
ChangeAccountInfo		38.94 s
ChangeAccountInfo.EditBio	passed	4.64 s
ChangeAccountInfo.EditName	passed	7.37 s
ChangeTheme		13.65 s
ChangeTheme.ChangeUserTheme	passed	3.09 s
SecretChat		13.45 s
SecretChat.newSecretChat	passed	4.06 s
NormalChat		2 m 21 s
NormalChat.CropBeforeSendImage	passed	15.12 s
NormalChat.FindUserAndCheckinbox	passed	6.40 s
NormalChat.SendEmoji	passed	9.81 s
NormalChat.SendImage	passed	6.01 s
NormalChat.SendMessage	passed	6.05 s
NormalChat.SendSticker	passed	7.79 s
NormalChat.ShareLocationMessage	passed	8.13 s
NormalChat.InHomePage	passed	2.06 s

Hình 3.18. Kết quả thực thi trên Android Device

Thực thi các ca kiểm thử trên Windows, kết quả trả về thành công trong thời gian 5.73s.

Appiumtests.DesktopApp: 3 total, 3 passed		5.73 s
com.intelij.rt.testing RemoteTestNGStarter -usedefaultlisteners false -socketfd4507 @w@C:\Users\DELL\AppData\Local\Temp\idea_working_dirs_testing\tmp C:\Users\DELL\AppData\Local\Temp\idea_testing\tmp		
Suite tests run: 1, Passes: 1, Failures: 0, Skips: 0		Process finished with exit code 0
Automation-Testing-main		Collapse Expand
TelegramTestWindow		5.73 s
TelegramTestWindow.TestChangeTheme	passed	1.64 s

Hình 3.19. Kết quả thực thi trên Windows Desktop

3.5. Kiểm thử trên nhiều nền tảng với Appium Grid:

Để thực hiện kiểm thử song song trên nhiều thiết bị, nhóm chọn sử dụng Appium kết hợp Selenium Grid (hướng dẫn cài đặt đã được hướng dẫn trong mục 2.6).

Cấu hình Appium Node

- Tạo tệp tin cấu hình dạng json cho từng thiết bị

```
{
  "capabilities": [
    {
      "deviceName": "AGC00007527",
      "platformVersion": "10.0",
      "maxInstances": 1,
      "platformName": "ANDROID"
    }
  ],
  "configuration": {
    "cleanUpCycle": 2000,
    "timeout": 30000,
    "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
    "maxSession": 1,
    "register": true,
    "registerCycle": 5000,
    "hubPort": 4444,
    "hubHost": "127.0.0.1",
    "hubProtocol": "http"
  }
}
```

Hình 3.20. Appium Grid: Tệp cấu hình cho thiết bị 1

```
{
  "capabilities": [
    {
      "deviceName": "177bd84f",
      "platformVersion": "8.1.0",
      "maxInstances": 1,
      "platformName": "ANDROID"
    }
  ],
  "configuration": {
    "cleanUpCycle": 2000,
    "timeout": 30000,
    "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
    "maxSession": 1,
    "register": true,
    "registerCycle": 5000,
    "hubPort": 4444,
    "hubHost": "127.0.0.1",
    "hubProtocol": "http"
  }
}
```

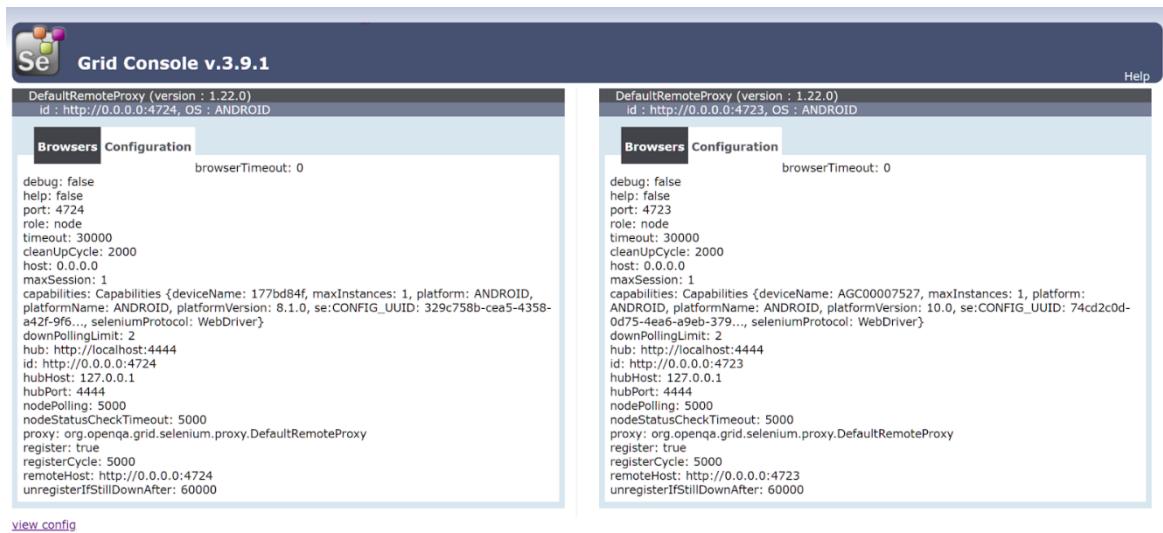
Hình 3.21. Appium Grid: Tệp cấu hình cho thiết bị 2

- Khởi tạo Appium Node với cổng 4723 cho thiết bị thứ nhất, cổng 4724 cho thiết bị thứ hai, tương ứng với hai tệp cấu hình đã tạo.
- > ***appium -p 4723 --nodeconfig Automation-Testing/src/test/node1.json***
- > ***appium -p 4724 --nodeconfig Automation-Testing/src/test/node2.json***

Thành công, command prompt của hub sẽ hiện thông báo như sau:

```
19:32:05.432 INFO - Selenium Grid hub is up and running
19:32:05.435 INFO - Nodes should register to http://192.168.1.9:4444/grid/register/
19:32:05.436 INFO - Clients should connect to http://192.168.1.9:4444/wd/hub
19:34:13.898 INFO - Registered a node http://0.0.0.0:4723
19:34:20.430 INFO - Registered a node http://0.0.0.0:4724
```

- Truy cập <http://localhost:4444/grid/console> để kiểm tra cấu hình thiết bị.



Viết ca kiểm thử song song

- Thiết lập Desired Capabilities cho từng thiết bị giống như quy trình kiểm thử thông thường.
- Chạy ca kiểm thử cho từng thiết bị, nếu chạy với nhiều threads thì sẽ kiểm thử song song trên các thiết bị, nếu không sẽ chạy lần lượt từng thiết bị.

```

    @Test
    public void openAccountInfoFirst() {
        DesiredCapabilities capabilities = new DesiredCapabilities();

        capabilities.setCapability(capabilityName: "platformName", jsonEnvironmentVariable.getString(key: "platformName"));
        capabilities.setCapability(capabilityName: "deviceName", jsonEnvironmentVariable.getString(key: "deviceName"));
        capabilities.setCapability(capabilityName: "app", APP);
        capabilities.setCapability(capabilityName: "automationName", jsonEnvironmentVariable.getString(key: "automationName"));
        capabilities.setCapability(capabilityName: "appWaitActivity", jsonEnvironmentVariable.getString(key: "appWaitActivity"));
        capabilities.setCapability(capabilityName: "noReset", jsonEnvironmentVariable.getBoolean(key: "noReset"));
        capabilities.setCapability(capabilityName: "platformVersion", jsonEnvironmentVariable.getString(key: "platformVersion"));

        try {
            driver = new AndroidDriver(new URL(APPIUM), capabilities);
            openAccountInfo(driver);
        } catch (WebDriverException | MalformedURLException e) {
            System.out.println("Cannot get your device");
            e.printStackTrace();
        }
    }
}

```

Hình 3.22. Testcase cho kiểm thử song song (1)

```

    @Test
    public void openAccountInfoSecond() {
        DesiredCapabilities capabilities = new DesiredCapabilities();

        capabilities.setCapability(capabilityName: "platformName", jsonEnvironmentVariable.getString(key: "platformName2"));
        capabilities.setCapability(capabilityName: "deviceName", jsonEnvironmentVariable.getString(key: "deviceName2"));
        capabilities.setCapability(capabilityName: "app", APP);
        capabilities.setCapability(capabilityName: "automationName", jsonEnvironmentVariable.getString(key: "automationName"));
        capabilities.setCapability(capabilityName: "appWaitActivity", jsonEnvironmentVariable.getString(key: "appWaitActivity"));
        capabilities.setCapability(capabilityName: "noReset", jsonEnvironmentVariable.getBoolean(key: "noReset"));
        capabilities.setCapability(capabilityName: "platformVersion", jsonEnvironmentVariable.getString(key: "platformVersion2"));

        try {
            driver = new AndroidDriver(new URL(APPIUM), capabilities);
            openAccountInfo(driver);
        } catch (WebDriverException | MalformedURLException e) {
            System.out.println("Cannot get your device");
            e.printStackTrace();
        }
    }
}

```

Hình 3.23. Testcase cho kiểm thử song song (2)

Kết quả thực thi kiểm thử

Automation-Testing	38.20 s
SecondDeviceTest	15.57 s
SecondDeviceTest.openAccountInfoSecond	passed 15.35 s
FirstDeviceTest	22.64 s
FirstDeviceTest.openAccountInfoFirst	passed 22.11 s

Hình 3.24. Kết quả kiểm thử song song

4. Tổng kết

Tóm lại, Appium đã giúp mô phỏng các hoạt động tương tác thủ công với ứng dụng di động để có thể thực hiện hiệu quả việc kiểm tra tự động hóa các ứng dụng. Với việc nhắm vào tự động hóa kiểm thử giao diện người dùng, chức năng và quy trình nghiệp vụ, Appium đã thực sự giải quyết vấn đề về cách cải thiện chất lượng ứng dụng di động và trải nghiệm người dùng cũng như cách sử dụng hợp lý các công cụ kiểm thử tự động, giúp tránh tiêu tốn công sức một cách lãng phí.

Các ứng dụng ngày nay đang được phát triển để hỗ trợ nhiều nền tảng khác nhau, và với mỗi môi trường cài đặt khác nhau, ứng dụng có thể xuất hiện những lỗi và sự cố chưa được phát hiện trong quá trình phát triển. Do đó, đòi hỏi việc kiểm thử phải được thực hiện đa dạng trên các nền tảng khác nhau. Và với điểm mạnh nhất là tính linh hoạt, hỗ trợ đa nền tảng, đa ngôn ngữ, Appium đã chứng minh nó là một công cụ kiểm thử vượt trội, một giải pháp thuận tiện giúp kiểm thử trở nên dễ dàng hơn.

Dẫu vậy, Appium có một số hạn chế trong hỗ trợ một số chức năng kiểm thử hay phiên bản thiết bị và yêu cầu người kiểm thử phải có hiểu biết về một ngôn ngữ lập trình cơ bản.

5. Tài liệu tham khảo

- [1] Manoj Hans. “*Appium Essentials*”, April 2015
- [2] Edureka Blog <https://www.edureka.co/blog/appium-architecture/>
- [3] Digital AI Blog <https://digital.ai/catalyst-blog/what-is-appium-server-a-complete-end-to-end-guide>
- [4] Wang Junmei, Wu Jihong (2019). “*Research on Mobile Application Automation Testing Technology Based on Appium*”. In International Conference on Virtual Reality and Intelligent Systems (ICVRIS).
- [5] Ma Xiaoquan. “*Design and Implementation of Mobile Automation Testing System Based on Appium*” [D]. Southeast University, 2018. (in Chinese)
- [6] Appium Docs <https://appium.io/docs/en/advanced-concepts/parallel-tests>