GUI AUTOMATION

Ashish Gurram
Electronics and Communication
Ramaiah Institute of Technology
Bengaluru, India
ashishgurram123@gmail.com

Srishti Agrawal
Computer Science
Ramaiah Institute of Technology
Bengaluru, India
srishtiagrawal60@gmail.com

Kumar Satyam
Information Science
Ramaiah Institute of Technology
Bengaluru, India
ks9430565@gmail.com

Anurag Paliwal
Electronics and Communication
Ramaiah Institute of Technology
Bengaluru, India
anuragpaliwal369@gmail.com

***Abstract*—Software testing can help you reduce errors, save maintenance costs, and lower overall software expenditures. In the software development life cycle, testing has become the most critical parameter (SDLC). Testing automation technologies make it simple for developers and testers to automate the complete software testing process. Engineers employed automated GUI (graphical user interface) testing methods to see if the program GUI appeared appropriately on different cellphones. We provide a GUI testing tool in this work that seeks to make it easier to test Android applications without sacrificing test accuracy. Furthermore, this program performs the entire testing process without human input, including creating test cases, mimicking user gestures, and confirming outcomes.New apps were chosen for our research not only because they are popular, but also because they support the most common user movements, such as tap, scroll, spread, and pinch. To examine the efficiency of this technology, we used five commercial Android phones and one popular news app in our trial. The results of our experiments suggest that our tool outperforms existing methods for determining whether a test has passed or failed. This paper discusses how to create a desktop application for automating visual gui testing using the Appium framework and Java in the NetBeans IDE for desktop applications. Testing automation technologies make it simple for developers and testers to automate the complete software testing process. Appium, a selenium-based framework (Selenium is a software tool which works with different browsers, operating systems, various programming languages etc).**

***Index Terms*—GUI Testing, Test Case, Automated Testing, Event Testing, Automated Testing, Visual GUI Testing, APPIUM, UI INSPECTOR**

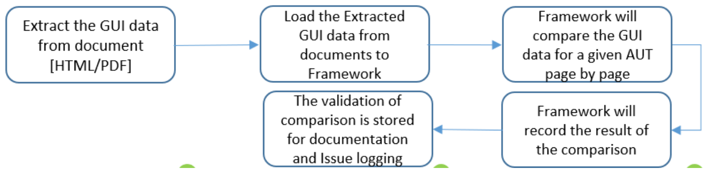
I. Introduction

The most important aspect of any software is the testing of the graphical user interface (GUI). This test was performed manually in the previous decade, therefore with each new revision of software, all tests must be performed again if a comprehensive functioning check is a need for the project, which is or should be in the majority of cases. Repeating these activities after each iteration is a time-consuming task that results in product delays as well as an increase in expense and work. Because the tester may not be familiar with software development, manual testing can result in human errors and poor results. As a result, automation technologies are being utilised to test the functionality of software. To perform GUI testing, these automated tools should contain the following fundamental features:
1) Users contributions can be recorded and played back using the record and play tool (some basic example like click on button, entering value in textbox box, selecting parameter from combo box etc).
2) Capturing data and images from a screen and being able to compare, store, and understand their attributes.
3) Compile and run the script, and after it has completed, be able to show the results neatly. This functionality is available in many testing software packages, along with many others such as object spy, insight object, error handling, and so on.
We create a window-based application that may be used for gui testing in this study. To create the desktop application, we'll use a demo application.
With the maturation of mobile internet services, the number of individuals utilising mobile applications has expanded all around the world.

Furthermore, automated GUI (Graphical User Interface) testing tools are highly needed in order to save software development time and cost. Automated GUI testing entails mimicking user gestures such as tap, long press, and scroll, and validating the resulting changes in the user interface, which is a mechanism for assessing if an app displays appropriately.
However, due to varying screen aspect ratios (the width-to-height ratio), the same text content in a news app may appear differently on different smartphones, such as breaking a line in a different location.
Our app will extract element details from the mobile application automatically and compare them to the json values from the testing dataset.

II. High Level Approach

A. Maintaining the Integrity of the Specifications

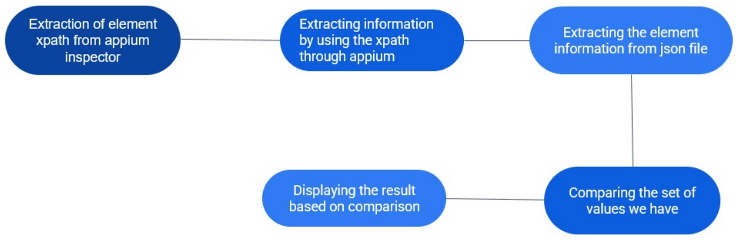


Extract the GUI data from document [HTML/PDF]
Load the Extracted GUI data from documents to Framework
Framework will compare the GUI data for a given AUT page by page
Framework will record the result of the comparison
The validation of comparison is stored for documentation and Issue logging

Fig. 1. High level approach flow chart

1) The main application is made by using java awt, java swing.
2) The connectivity of mobile application is done by using Appium.
3) And used Appium inspector to obtain the gui information of the gui elements present in different pages.
4) Application data should be there in json format for comparison purposes.
5) And comparing the json values of application with the extracted values.
6) And logging in the results in text format.

III. Proposed System.



Extraction of element xpath from appium inspector
Extracting information by using the xpath through appium
Extracting the element information from json file
Comparing the set of values we have
Displaying the result based on comparison

Fig. 2. Proposed system flow

1) Desktop application is made for the purpose of automating gui testing which will reduce the time consumption of testing the apps and also will be more efficient then doing it manually.
2) Our project is based on java, where java is used as are primary language integrated with Appium framework.
3) Appium framework is basically used to automate processes of mobile phones, i.e writing code using Appium will allow us to automate basic mobile processes like clicking, scrolling and double tapping.
4) Starting with the problem statement we started off by connecting the testing device to our systems. For which we need Appium server running in the back.
5) We connected the mobile application activity with Appium server by giving the UDID, appactivity, apppackage, name of the device under test, platform version and platform name.
6) Data pre-processing is done to extract specific attributes of the required elements from the json files of the mobile application under test.
7) And then we extracted all the xpaths of the gui elements from the application under test using the feature given under Appium called as Appium inspector.
8) Appium inspector is basically used for inspection of gui elements which is a standard procedure to identify the ui elements of a mobile application uniquely.
9) Using the xpaths of the gui elements now we can control the elements(i.e clicking,scrolling etc) by using our developed application.
10) Now by using these xpaths itself we are extracting all the needed attributes of the application under test by developing a algorithm for that specific attribute we are trying to extract.
11) Then we extract the required attributes values from the json files of the application under test.
12) Now these two sets of data are compared for likeliness to check if the output of the screen is matching with the desired values.
13) If the values match the output is shown as pass.
14) And if the values don't match the output is shown as fail.
15) And all the results are logged into a text document for analysis by the testing team.
16) And all of these things are integrated inside a desktop application.
17) Which is created by integrating java with Appium.
18) So this is how can test gui of the applications efficiently and also saving a lot of time and effort.

Future Works

1) Adding OCR to extract text from images.
2) Automating the whole extraction of xpaths from the elements.
3) Extraction of font, font size and font color.
4) Have to work for different screen resolutions.

References

[1] Ruiz and Y. W. Price, "GUI Testing Made Easy," Testing: Academic Industrial Conference - Practice and Research Techniques (taic part 2008), 2008, pp. 99-103, doi: 10.1109/TAIC-PART.2008.11.
[2] E. T. -. Chu and J. Lin, "Automated GUI Testing for Android News Applications," 2018 International Symposium on Computer, Consumer and Control (IS3C), 2018, pp. 14-17, doi: 10.1109/IS3C.2018.00013.
[3] Ashwaq A Alotaibi, M. Rizwan Jameel Qureshi "Novel Framework for Automation Testing of Mobile Applications using Appium",February 2017,International Journal of Modern Education and Computer Science 9(2):34-40,DOI:10.5815/ijmecs.2017.02.04
[4] https://www.semanticscholar.org/paper/Software-Testing-Automation-using-Appium-Shah-Shah/2e7cdfe3e64575401901cb3867378bf793db2d03
[5] https://ieeexplore.ieee.org/document/8920799
[6] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1049.9945rep=rep1type=pdf
[7] https://appium.io/docs/en/drivers/android-uiautomator/
[8] https://appium.io/docs/en/about-appium/intro/
[9] https://iarjset.com/upload/2017/si/NCIARCSE-2017/IARJSET-NCIARCSE
[10] https://www.researchgate.net/publication/257391348 _Graphical_user_interface_GUI_testing_Systematic_mapping_and_repository