Zhengkun Lou

EECS 4443 Assignment Report

March/19/2023

Part 1a: Automated UI Testing

Main Cases:

Test Script ID	004
Title	User want to Next page/Check famous person information/Find by name
Test Case ID	/Users/zklou/Desktop/A1/Demo_Quotation/app/src/androidTest/java/ca/yorku/eecs/mack/demoquotation/DemoQuatationActivityTest004
Test Setup	
Data	
Procedure	Start the App next page: Click on main screen area to next page Check information: Swap to left/right/top/button on main area Find by name: Press and hold for 3 seconds on main screen Select specific person by scroll up/down on the list
Actual Result	Change to next "famous person picture" with quotation Change the picture into introduction Change the person to the aim person
Status	Pass/Pass/Pass
Defect Cross-Reference	

Test Script ID	005
Title	User want to do quiz
Test Case ID	/Users/zklou/Desktop/A1/Demo_Quotation/app/src/androidTest/java/ca/yorku/eecs/mack/demoquotation/DemoQuatationActivityTest005
Test Setup	
Data	
Procedure	Start the App Click the button on right upper corner Click Quiz Swipe for hint at question area (left right Click the choice box to make choice
Actual Result	Change the question into hint and answer the question
Status	Pass
Defect Cross-Reference	

Test Script ID	006
Title	Settings
Test Case ID	/Users/zklou/Desktop/A1/Demo_Quotation/app/src/androidTest/java/ca/yorku/eecs/mack/demoquotation/DemoQuatationActivityTest006
Test Setup	
Data	
Procedure	Start the App Click the button on right upper corner Click setting Change "Quiz length", turn on/off "Hints/vibration/audio"
Actual Result	Change "Quiz length", turn on/off "Hints/vibration/audio"
Status	Pass
Defect Cross-Reference	User cannot set a custom amount of questions

Part 1b: Add a new search functionality and test it.

In Demo-Quotation, I added the search menu time in the application's context menu. This was because I found that a HELP function was not being implemented. While I was adding the search function, I used the "HELP" function as a basis for the search, since I believe that search is a type of HELP. Therefore, the name "HELP" remained unchanged. To implement the search function, I added SearchActivity.java, Menu.XML, and Activity_Search.XML.

Fully functional application could be found under App folder.

Test Script ID	008
Title	User want to do 20 questions quiz and before that user want to search for quote, name, information.
Test Case ID	Demo_Quotation/app/src/androidTest/java/ca/yorku/eecs/mack/demoquotation/DemoQuatationActivityTestMain
Test Setup	Have key-word stored in String.xml
Data	Lincoln-“Name”/Vote-”Content Of Quotes”/ wounded-”Self-Introduction:”
Procedure	Start the App Click the button on right upper corner Set question to 20, add hints, vibration, audio. Back to main page, select HELP to search. Back to main page, select Quiz to start. Swipe for hint at question area (left right Click the choice box to make choice
Actual Result	Change the question into hint and answer the question
Status	Pass
Defect Cross-Reference	User cannot use search function while doing quiz

Part 2: Profile your application

Profile Result: Pictures can be found under Profile folder

Title	Memory Usage(Mb)	CPU Usage(%)+	Energy Usage
Test ID:011 do click swap	88	13%	Light
Test ID:012 do settings: turn on/turn off	90	11%	Light
Test ID:013 do searches, keyword: ”Lincoln” ,”Vote” , “wounded”	99.9	17%	Light
Test ID:014 do 20 questions with all set/audio/ vibration and Hints	122.9	13%	Light
Test ID:015 do 10 questions with all setting on: audio/ vibration Hints	79	27%	Light
Test Environment: PC: CPU:3.3G Xeon E5 GPU:1080 8GB MEMO:32GB Emulator: Test on Nexus 5X API 30 Android version 11			

Part 3: Benchmark your application

Based on the Part 2 results, the two use cases with the highest utilization in memory and CPU are 013 and 014. As all cases have light energy usage, using 013 and 014 is sufficient for energy usage instead of picking more cases. To test cases 011 and 014, we could set up the micro benchmark with input string =4 and running circle =500.

013:
Methodology:
In order to micro benchmark the settings menu, I plan to create a Clicker.java file and define the necessary function. The clicker will be set up to randomly select child positions 1, 2, and 3. Additionally, we will define the clicker with varying frequencies, such as 1 operation per sec, 1 operation per 2 sec and keep increase the time, to simulate different user clicking speeds. This will allow us to benchmark the functionality of the settings menu in different scenarios. Then I will analyze the results to gain insight into the performance of the settings menu under different pressure.

014:
Methodology:
To implement this approach, I need to create a text file and store some words as a test file. I will then create a method in MicrobenchmarkBuilder.java to define the string we imported as plain alphabetical. This will enable us to analyze the search function under different conditions. After defining the method, I will add the necessary dependencies into the build.gradle file. Then test, run, and debug the function to ensure that it is working correctly. Finally, I will analyze the results to gain insight into the performance of the search function under specific words volume.

Conclusion:
Overall, the micro benchmarking will help us to test the performance of in high usage cases which are “013:settings” and “014:search” and find out the limit value of Demo_quotation. For setting, I do the pressure test by increase/ decrease operation frequent. For search, I do the pressure test by adding/deleting text case numbers. To finalize the results, we could analysis our application in performance and identify any areas for improvement.