FIT3036 Computer Science Project

MONASH University

Test Report

TNK - The Numeric Keyboard

Design of keyboards and other interfaces for users with reduced capacity

Student Name	: Reynald Nixon
Student ID	: 27022722
Supervisor Name	: Rasika Amarasiri
Word Count	: 2230

Table of Contents

1.0. Introduction

The objective of this report is to deliver the result of the testing done for the TNK - The Numeric Keyboard. The test to be done are unit testing and system testing that will be further explained in section 2. As described in the test plan of the project proposal, testing will be done by myself and other people testing the program in the development. Initially, the unit testing will be done by using Mocha and Chai for JavaScript but due to problems that occur during the project development such as other unit's assignments and laptop that broke down during the development phase, testing using software was abandoned and instead, testing was done manually. The approach of the testing will be entering the input of key combination into the demo text editor and see if it produces the correct output. Another method to be use will be putting a log in the console to see whether the log produced is correct. The data to be collected will be whether the actual results matches the expected results and will be in a list of passes or fails. Results will then be presented in the format of tables. Test scenario done in this report will be based from the design specifications and requirements of the project.

In the project proposal, there are no specific test requirement that needs to be verified in this test report as the testing implementation of the program is done during the entire development phase. There were 2 sample test cases in the test plan before which is testing the key combination test case and the word suggestion feature test case. The key combination test case will be done below, however, the word suggestion feature test case will not be found in this report because the word suggestion feature was failed to be implemented in the final product.

As the project was done in a slight amount of time with problems occurred, the test target will be 75% success rate for each scenario before moving to a different scenario. An example will be for the key combination test case, there were 39 key combinations in the final product. For the scenario to be called successful, at least 30 key combination needs to have the correct output for that scenario to pass the test scenario. The test coverage will focus on the mapping of the keyboard since it's the main element of the project therefore the code that will be covered most is the mapping part. However, the test will also cover some UI related problem and the easiness of use for the program.

The main criteria to decide that a testing phase is complete for a scenario will be if a function successfully does what it supposed to do. For example, a keyboard mapping function test will be considered as complete if 75% of the numbers combination entered maps into the correct characters or function in the text editor because the keyboard mapping function was supposed to map number combinations into characters or functions. Side criteria for the testing will be if the whole system meets the project requirements of the proposal. Example of it will be if it meets the functional requirements which is creating a keyboard layout to adapt users with one hand to interact with the keyboard efficiently or if the project meets the non-functional requirements that was proposed as in if the project has documentations, is it efficient and effective, etc.

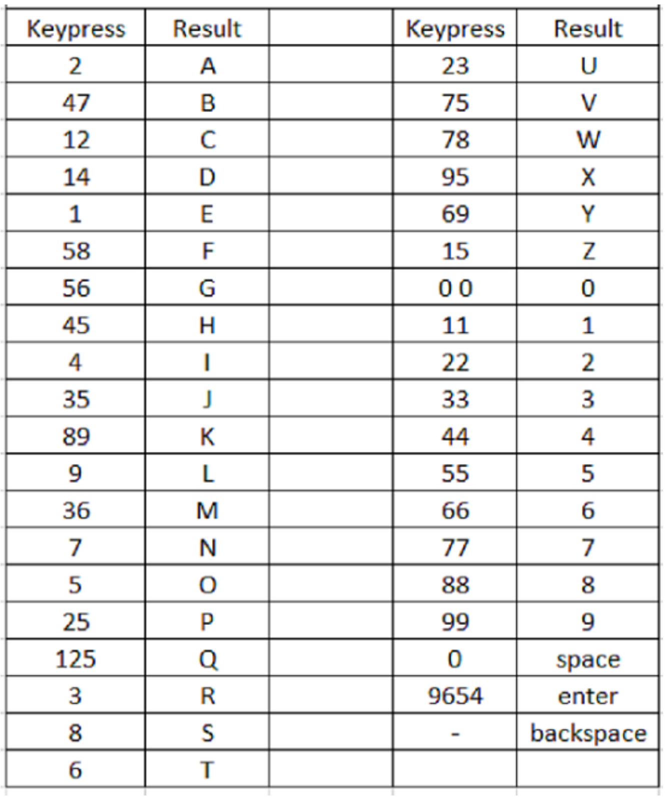
2.0. Test Report

The testing to be done will be unit testing for individual functions and system testing for the whole function. As described in the proposal, unit testing will focus mainly on testing the whether the numeric key combinations mapped to the expected letters successfully. The system testing will cover the acceptance testing, performance testing, usability testing, destructive/stress testing and alpha testing to a few of my colleagues. The aim of the test is to achieve at least 75% successful rate for each of the test to be considered accepted.

2.1. Unit Testing

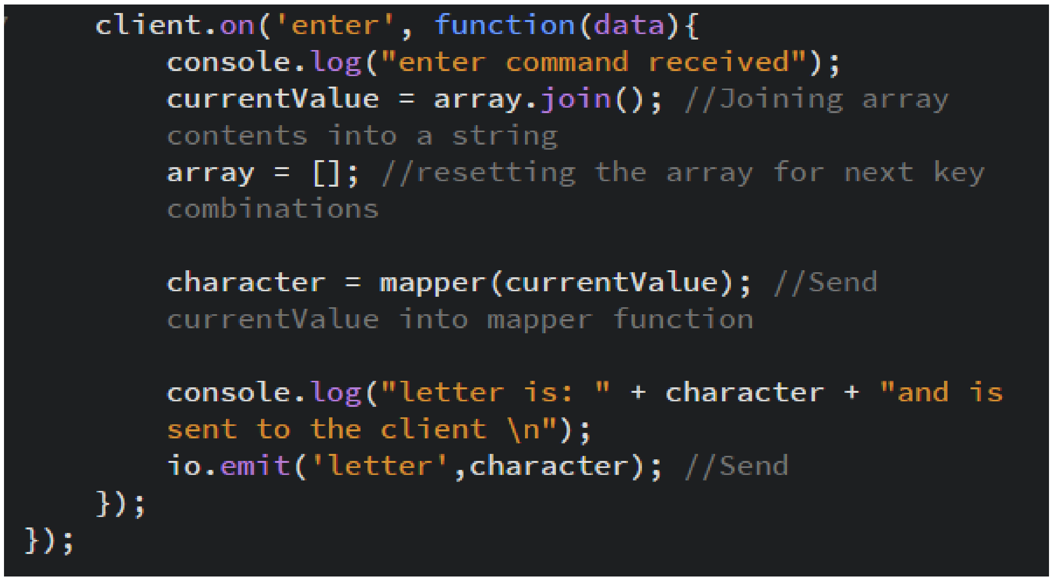
2.1.1. Key combination/Mapper function test case

The test to be done here is whether the input of key combination by user maps to the expected letter that is provided in the picture below (Figure 1). This test is targeted to test the mapper function in the app.js. The outcome of the test is provided in a test case format of tables. The table contains the scenario, steps, expected and actual results, also the pass or fail of the scenario. This testing is necessary to ensure that the key combinations produces the correct output so that user can use the product.



Keypress	Result		Keypress	Result
2	A		23	U
47	B		75	V
12	C		78	W
14	D		95	X
1	E		69	Y
58	F		15	Z
56	G		0 0	0
45	H		11	1
4	I		22	2
35	J		33	3
89	K		44	4
9	L		55	5
36	M		66	6
7	N		77	7
5	O		88	8
25	P		99	9
125	Q		0	space
3	R		9654	enter
8	S		-	backspace
6	T			

Figure 1. Cheat sheet of the key combinations available



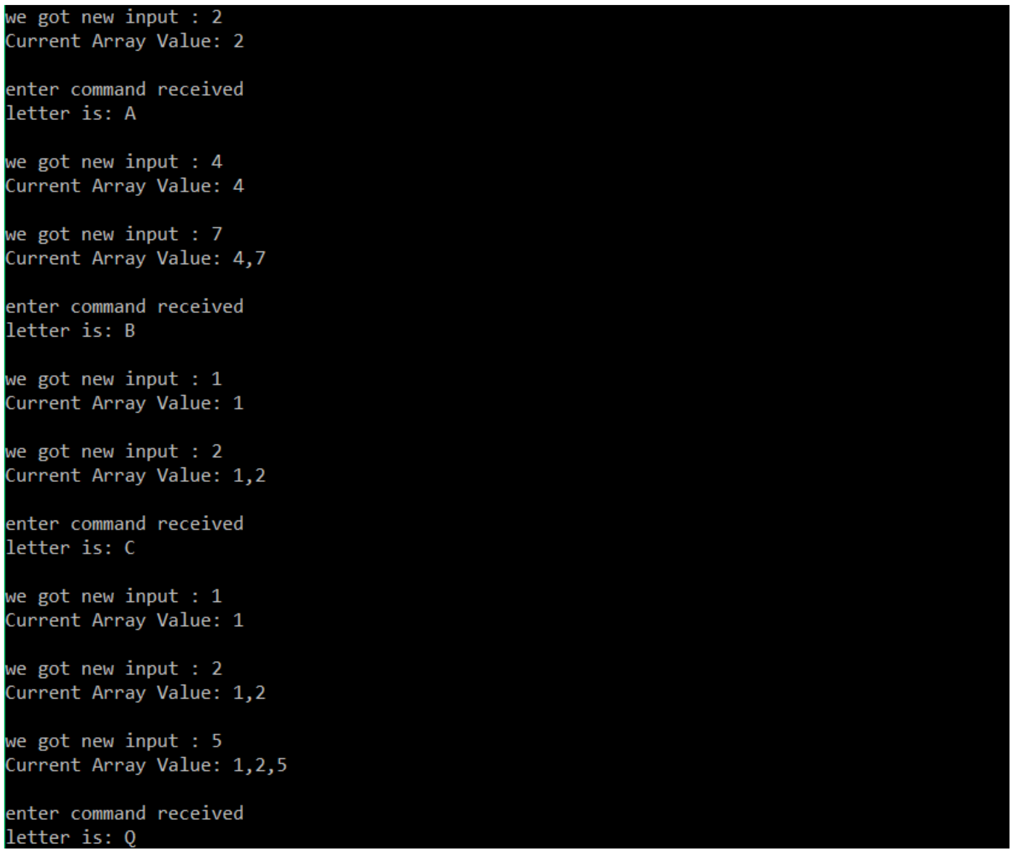
```
    client.on('enter', function(data){
        console.log("enter command received");
        currentValue = array.join(); //Joining array
        contents into a string
        array = []; //resetting the array for next key
        combinations

        character = mapper(currentValue); //Send
        currentValue into mapper function

        console.log("letter is: " + character + "and is
        sent to the client \n");
        io.emit('letter',character); //Send
    });
});
```

Figure 2. Part of code in app.js that user mapper function

Case ID	Scenario	Steps	Data	Expected Results	Actual Results	Pass/Fail
A1	Correct key combination entered to the mapper function	1. Enter correct key combination into the mapper function 2. Mapper will convert key combination into a letter 3. See result of letter produced if it matches figure 1	Correct key combinations in figure 1 is send into the mapper function	The console log will display combinations entered and the corresponding letter it produces that matches figure 1	Every key combination maps to the corresponding letter or function as displayed in the console according to figure 1	PASS

A2	Wrong key combination entered to the mapper function	1. Enter wrong key combination into the mapper function 2. Mapper will interpret key combination 3. See result of interpretation in console log	User enters the key combination that is not available in figure 1	Console log will display combinations entered and display blank on letter produced in console log	Entering different key combination that are not available in figure 1 results in the console displaying the combinations entered and displays blank in letter produced	PASS

Figure 3. Test case for mapper function



```
we got new input : 2
Current Array Value: 2

enter command received
letter is: A

we got new input : 4
Current Array Value: 4

we got new input : 7
Current Array Value: 4,7

enter command received
letter is: B

we got new input : 1
Current Array Value: 1

we got new input : 2
Current Array Value: 1,2

enter command received
letter is: C

we got new input : 1
Current Array Value: 1

we got new input : 2
Current Array Value: 1,2

we got new input : 5
Current Array Value: 1,2,5

enter command received
letter is: Q
```

Figure 4. Console log for mapper function

As seen in Figure 3, this test achieves 100% success rate and it passes the criteria of at least 75% success rate of correct letter mapping. For test A1, every 39 possible key combinations in figure 1 has been tried by following the steps and the result is correct letter or function produced in console log. For test A2, by trying different key combinations that are not available in figure 1, it produces the expected result every time hence passing the test as well. An example of the console log is also provided above in Figure 3, in which we can see that current array value equals 2 and when enter command is received, the letter is A. same thing goes with 4 and 7 which results in B and 1,2,5 which corresponds to Q.

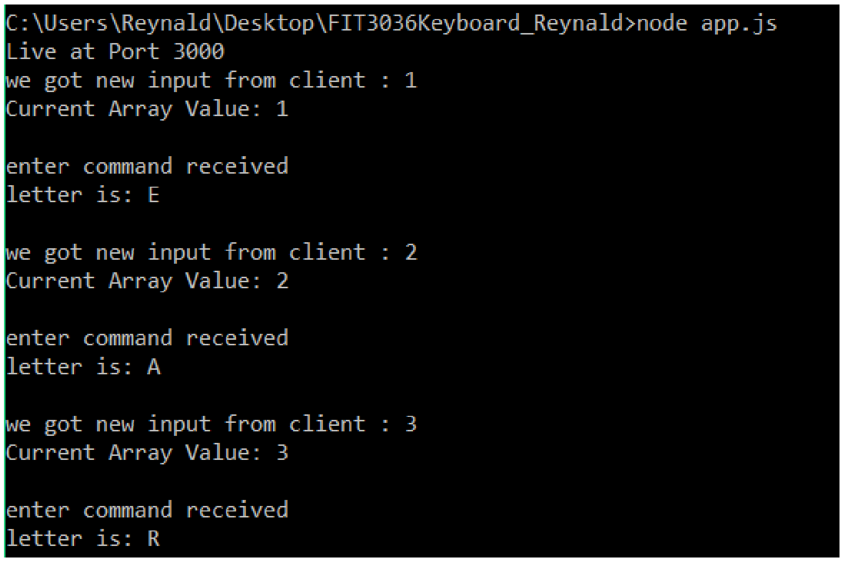
2.1.2. Socket.io Test Case

The test to be done here is whether the socket.io module successfully receives any data from the client and whether the server successfully sends any data to the client both in real-time. The outcome of the test is provided in a test case format of tables. The table contains the scenario, steps, expected and actual results, also the pass or fail of the scenario.

Case ID	Scenario	Steps	Data	Expected Results	Actual Results	Pass/Fail
B1	Client sends data to server in real-time	1. Enter any input in the client side that invokes socket.io function to work 2. See console whether it receives any data from client because the log is produced by server	Send any input from client that invokes socket.io to send data to server	The console log will display the input received from the client side	Console log displays the input received from client correctly	PASS

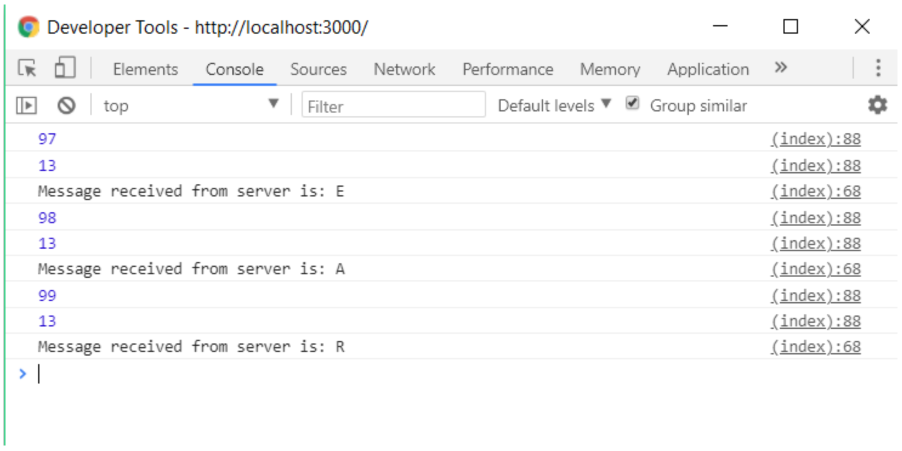
B2	Server sends data to client in real-time	1. Enter any input in client side that invokes socket.io 2. See console whether it sends any data from server side to client	Send any input from server that invokes socket.io to send data to client	Client displays the input send from server	Client displays the input send from server correctly	PASS

Figure 4. Test case for socket.io module



```
C:\Users\Reynald\Desktop\FIT3036Keyboard_Reynald>node app.js
Live at Port 3000
we got new input from client : 1
Current Array Value: 1

enter command received
letter is: E

we got new input from client : 2
Current Array Value: 2

enter command received
letter is: A

we got new input from client : 3
Current Array Value: 3

enter command received
letter is: R
```

Figure 5. Console log on server-side displays message received from client



```
97                                        (index):88
13                                        (index):88
Message received from server is: E        (index):68
98                                        (index):88
13                                        (index):88
Message received from server is: A        (index):68
99                                        (index):88
13                                        (index):88
Message received from server is: R        (index):68
>
```

Figure 6. Console log on client-side displays message received from server

As seen in Figure 4, this test is successful because socket.io successfully send data from client to server and vice versa in real-time which is the objective of implementing socket.io. For test B1, the console in app.js manage to display the input from the client correctly in real-time (Figure 5) and for B2, the console in the browser manage to display the input from the server correctly in real-time (Figure 6). Both indicates that Socket.io is working perfectly in sending messages between server and client in real time.

2.2. System Testing

2.2.1. Acceptance Testing

Acceptance testing is a level of software testing where a system is tested for acceptability ("Acceptance Testing - Software Testing Fundamentals", n.d.). It is to evaluate whether the system met its business requirements. The business requirement of the project is to design a new keyboard layout to adapt users with one-hand to use the keyboard efficiently to match users using QWERTY keyboard with 2 hands. The method of this testing is asking 5 of my colleagues by explaining the business requirement and explaining the project itself. The numeric keyboard is considered to pass this test as 5 of my colleagues agreed that the business requirement is met in which user can type efficiently in the numeric keypad because of its space that is reachable by 1 hand and the fingers do not need to travel in a far distance compared to one hand usage of QWERTY keyboard. It also meets its non-functional requirements as described in the project proposal which includes documentation, is efficient and effective, an open source project, scalable, high performance, robust and supports usability because it can be used by people with one hand only or people with two hands can use the text editor too using the normal QWERTY layout.

2.2.2. Performance Testing

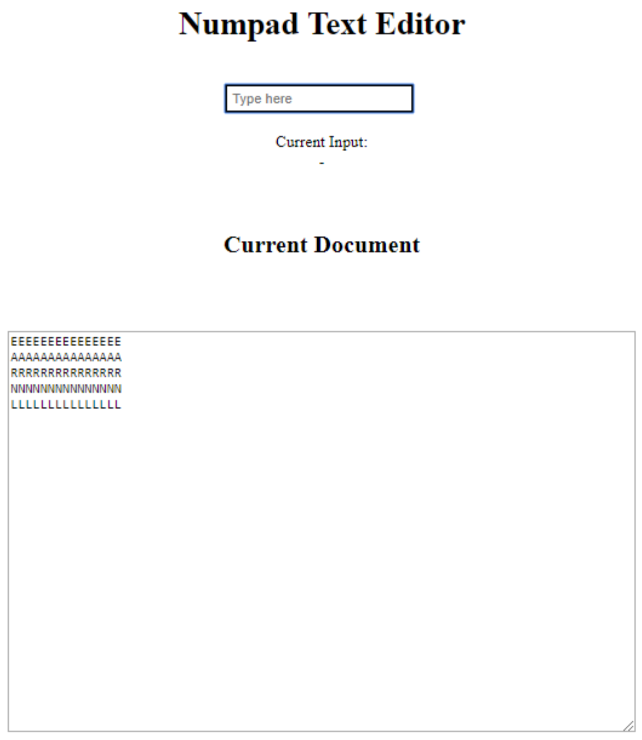
Performance testing is done to provide stakeholders with information about the application regarding speed, stability and scalability ("Performance Testing Tutorial: Types, Process & Important Metrics", n.d.). An average typist will type in the QWERTY keyboard layout with the speed of 40 WPM (words per minutes) (Karat, Halverson, Horn & Karat, 1999). The target of the project according to the proposal is to achieve a typing speed of at least 25 WPM. Based on the test on 5 of my colleagues and myself, the average WPM is 20. In this case, the project failed the performance test. The major cause of the slow typing speed is mainly because we are not used to the key combination and need to refer to the cheat sheet at most time, which takes quite a bit of time. If the user memorizes the key combination, it may increase the speed to achieve the initial target of 25 WPM.

2.2.3. Usability Testing

Usability testing involves testing the app on different users to see if it caters to different people. In this case, the project is considered to pass usability testing if it caters to not only one hand users, but to two hand users with the normal keyboard layout as well. In this manner, the project passes the usability testing because one hand user can use the project by typing in the numeric key combinations while normal users with two hands using QWERTY keyboard layout can use the text editor as well by directly typing in the current document section.

2.2.4. Stress Testing

Stress testing refers to the testing of software to determine whether its performance is satisfactory under any extreme and unfavourable conditions ("What is Stress Testing? - Definition from Techopedia", n.d.). The acceptance rate for the stress testing is if users input 75 key combinations continuously, the system will produce the correct output and not break down. The test method will be opening the project and putting in 75 key combinations rapidly and see if the document produces the correct output. The result of the test can be seen below in Figure 7 where it successfully passed the test.



Numpad Text Editor

Type here

Current Input:
-

Current Document

```
EEEEEEEEEEEEEEE
AAAAAAAAAAAAAAA
RRRRRRRRRRRRRRR
NNNNNNNNNNNNNNN
LLLLLLLLLLLLLLL
```

Figure 7. Rapid input into the project 75 times

2.2.5. Beta Testing

Beta testing is the tests that is meant for people that are not involved in the development phase for the project. Here the aim is to get at least 3 people to test the project and gain their feedback. The result is the project has been tested by at least 5 people and it passes the target of at least 3 people. The feedback gained from the testers also indicate that the app accomplished the business requirement and has potential to be a new keyboard layout for users with one hand.

2.2.6. Integration Testing

Integration testing is a level of software testing where individual units are combined and tested as a group ("Integration Testing - Software Testing Fundamentals", n.d.). The test method is mostly done by opening the text editor and type in all the possible input according to the key combination cheat sheet. After that we check the result and if it matches the cheat sheet, we consider it passes the test. It combines the mapper function and the socket.io module to work together as one app.

Case ID	Scenario	Steps	Data	Expected Results	Actual Results	Pass/Fail
C1	Users type in correct key combination in text box and corresponding letter will appear in text editor	1. Go to text editor webpage 2. Type in a key combination on the text box 3. Press enter 4. See the result after pressing enter	User enters the possible key combination that is available in figure 1	The corresponding letter or function will appear correctly in the current document according to the cheat sheet in figure 1	Every key combination maps to the corresponding letter or function in the current document according to the cheat sheet in figure 1	PASS

C2	Users type in wrong key combination in text box	1. Go to text editor webpage 2. Type in a key combination on the text box 3. Press enter 4. See the result after pressing enter	User enters the key combination that is not available in figure 1	Nothing will appear in the current document, also resets the current input section into -	Entering different key combination that are not available in figure 1 results nothing in current document and current input resets into -	PASS

Figure 8. Test case for integration testing of socket.io + mapper function

As seen in Figure 8, this test achieves 100% success rate and it passes the criteria of at least 75% success rate of correct letter produced in client side. For test C1, every 39 possible key combinations in figure 1 has been tried by following the steps and the result is correct letter or function produced the current document section. For test C2, by trying different key combinations that are not available in figure 1, it results in nothing produce in current document and the current input resets into "-" hence passing the test as well.

2.2.7. Function Testing

The Numeric Keyboard is meant to be an alternative keyboard layout for people with 1 hand to use the keyboard in an efficient matter. After testing the keyboard layout to 5 of my colleagues, they all agree that The Numeric Keyboard serves the function to cater to people with 1 hand as the hand does not need to travel in a far distance. The mapping is efficient as well as it is located in one place which is the numeric keypad section. The only concern that the testers give to developer was that to use the keyboard efficiently, they need to memorize the cheat sheet which is not easy. Overall, they were satisfied with the project.

3.0. Conclusion

This report discussed 2 major testing that was done for The Numeric Keyboard project which is the unit testing and system testing. In unit testing section, the modules that were tested are the mapper function in the server side and the socket.io module that connects the server and client also sending data between them in real-time. Both module passed the acceptance criteria for the unit testing. For the system testing section, various tests were done: acceptance testing, performance testing, usability testing, stress testing, beta testing, integration testing and function testing. In the system testing section, the project manages to pass all the test except in the performance test which was below the expected words per minutes of 25 WPM. The problem lies in the custom mapping that the TNK provides, which user will need to memorize in order to use the keyboard layout efficiently. Even though TNK did not meet the acceptance criteria of the performance test, it does not mean that TNK fails the test, it's because users have not memorized the key combination. Overall, TNK is considered to pass all the test because out of all the test, only performance test was not passed and it passes at least 75% of tests done.

4.0. Bibliography

[1] Acceptance Testing - Software Testing Fundamentals. Retrieved from http://softwaretestingfundamentals.com/acceptance-testing/

[2] Performance Testing Tutorial: Types, Process & Important Metrics. Retrieved from https://www.guru99.com/performance-testing.html

[3] Karat, C., Halverson, C., Horn, D., & Karat, J. (1999). Patterns of entry and correction in large vocabulary continuous speech recognition systems. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems (pp. 568-575). Pittsburgh, Pennsylvania, USA. Retrieved from https://doiorg.ezproxy.lib.monash.edu.au/10.1145/302979.303160

[4] What is Stress Testing? - Definition from Techopedia. Retrieved from https://www.techopedia.com/definition/15310/stress-testing

[5] Integration Testing - Software Testing Fundamentals. Retrieved from http://softwaretestingfundamentals.com/integration-testing/