Master of Science in Electrical Engineering with emphasis on

Telecommunication Systems

June 2019

PERFORMANCE TESTING AND ASSESSMENT OF VARIOUS NETWORK-BASED APPLICATIONS.

DIVYA NAGA KRISHNA KONDEPATI

SATISH KUMAR REDDY MALLIDI

Faculty of Computing, Blekinge Institute of Technology, 37139, Karlskrona

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Telecommunication Systems. This thesis is equivalent to 20 weeks of full-time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author(s):
Divya Naga Krishna Kondepati
Email: diko17@student.bth.se
Satish Kumar Reddy Mallidi
Email: samc17@student.bth.se

University advisor:
Prof. Dr-Ing Markus Fiedler
Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona,
Sweden.

Internet: www.bth.se
Phone: +46 455 38 50 00
Fax: +46 455 38 50 57

ABSTRACT

Performance Testing is one of the crucial parts of any software cycle process. In today's world, there is any number of network-based applications. Manual Testing and Automated Testing are the two important ways to test any type of application. For Manual Testing a mobile application known as BlekingeTrafiken is used. For Automated Testing, a web application known as Edmodo is used. Selenium is the automated tool included for automated testing. But, for each application, there are several users and because of that, there might be a decrease in performance of the application as an increase in the number of users. Performance of an application also depends on response times, mean, stability, speed, capacity, accuracy. The performance also depends on the device (memory consumption, battery, software variation) and Server/API (less no of calls) and depends on the network performance (jitters, packet loss, network speed). There are several tools for performance testing. By using these tools, we can get accurate performance results of each request.

In this thesis, we performed manual testing of a mobile application by increasing the number of users under similar network conditions, automated testing of a web application under various test cases and tested the performance of an iPad application (PLANETJAKTEN). It is a real-time gaming application used to learn mathematics for children. Apache JMeter is the tool used for performance testing. The interaction between the JMeter tool and the iPad is done through HTTP Proxy method. When any user starts using the application, we can measure the performance of each request sent by the user. Nagios is the tool used to monitor the various environments. Results show that for manual testing, the time taken for connecting to WI-FI is low compared to opening and using the application. For automated testing, it is found that the time taken to run each test case for the first time is high compared to the remaining trials. For performance testing, the experimental results show that the error percentage (the percentage of failed requests) is high for logging into the application compared to using the application.

ACKNOWLEDGEMENTS

Firstly, we would like to express our heartfelt gratitude to our supervisor Dr. Markus Fiedler for his constant support, encouragement, for generously sparing time and completed insight about the topic and providing valuable comments throughout research of this Thesis work accomplishing various tasks and composing the report. His patience and immense knowledge make him a great mentor and his helpfulness makes him a great human being.

We would like to thank Henrik Rosvall at Scientific EdTech at Lund for the application we used for our research. We are very thankful for the instructions and login details.

We would like to thank our parents for giving us moral and financial support. We also thank all friends, colleagues working with Prof. Markus and our dear classmates.

TABLE OF CONTENTS

Chapter 1 10
INTRODUCTION 10
1.1 MANUAL TESTING 10
1.2 AUTOMATED TESTING 11
1.3 PERFORMANCE TESTING 12
1.4 PROBLEM STATEMENT 12
1.5 RESEARCH QUESTIONS 13
1.6 METHODOLOGY 13
1.7 THESIS OUTLINE 13
Chapter 2 15
BACKGROUND 15
2.1 SELENIUM 15
2.2 APACHE JMETER 17
2.3 PLANETJAKTEN 18
2.4 NAGIOS 18
Chapter 3 20
RELATED WORK 20
Chapter 4 22
METHODOLOGY 22
4.1 INSTALLATIONS 22
4.2 EXPERIMENTAL SET UP - ARCHITECTURES 23
4.2.1 MANUAL TESTING 23
4.2.2 AUTOMATED TESTING 25
4.2.3 PERFORMANCE TESTING OF AN IPAD APPLICATION 27
Chapter 5 32
VALIDATIONS & EXPERIMENTAL PARAMETERS 32
5.1 TEST CASES 32
5.1.1 Test Cases for Manual Testing of Mobile Application 32
5.1.2 Test Cases for Automated Testing of a Web Application 33

5.1.3 Test Cases for Performance Testing of iPad Application 34
5.2 VALIDATION FOR PERFORMANCE TESTING 34
5.2.1 VALIDATION 1 36
5.2.2 VALIDATION 2 37
5.2.3 VALIDATION 3 38
Chapter 6 40
RESULTS & ANALYSIS 40
6.1 MANUAL TESTING OF MOBILE APPLICATION 40
6.2 AUTOMATED TESTING OF EDMODO APPLICATION 44
6.3 PERFORMANCE TESTING OF IPAD APPLICATION 45
6.3.1 VALIDATION 1 46
6.3.2 VALIDATION 2 49
6.3.3 VALIDATION 3 52
6.3.4 VALIDATION 4 55
6.3.5 COMPARISON OF ERROR PERCENTAGE FOR ALL VALIDATIONS 58
Chapter 7 60
RESEARCH QUESTIONS & ANSWERS 60
7.1 RESEARCH QUESTIONS & ANSWERS 60
Chapter 8 62
CONCLUSION & FUTURE WORK 62
8.1 CONCLUSION 62
8.2 FUTURE WORK 63
Chapter 9 64
REFERENCES 64
Chapter 10 66
APPENDIX 66
Appendix A 67
Test Case with network A, no background applications & Login 67
Average of Summary Report 67
Average of Aggregate Report 68
Test Case with network A, no background applications & Run the application 69
Average of Summary Report 69
Average of Aggregate Report 70

Appendix B 71
Test Case with network B, no background applications & Login 71
Average of Summary Report 71
Average of Aggregate Report 72
Test Case with network B, no background applications & Run the application 73
Average of Summary Report 73
Average of Aggregate Report 74
Appendix C 75
Test Case with network A, 4 background applications & Login Average of Summary Report 75
Average of Aggregate Report 76
Test Case with network A, 4 background applications & Run the application 77
Average of Summary Report 77
Average of Aggregate Report 78
Appendix D 79

LIST OF FIGURES

Figure 2.1 Selenium 15
Figure 2.2 Element locators in Selenium 16
Figure 2.3 Architecture of Selenium web driver 17
Figure 2.4 Architecture of Apache JMeter 18
Figure 2.5 Architecture of Nagios 19
Figure 4.1 Installation of Selenium 22
Figure 4.2 Experimental Setup for manual testing 23
Figure 4.3 Experimental Setup for Automation testing 25
Figure 4.4 Architecture of Automated Testing for Edmodo Application 26
Figure 4.5 Experimental Setup for Performance testing 27
Figure 4.6 Process for Root Configuration 29
Figure 4.7 Architecture of JMeter for iPad application 29
Figure 4.8 Architecture of Nagios in our Project 30
Figure 4.9 Configuration of MobiosPush 30
Figure 4.10 Nagios Dashboard 31
Figure 5.1 Validation 1 with network A & no background apps 36
Figure 5.2 Validation 2 with network B & no background apps 37
Figure 5.3 Validation 3 with network A & 4 background apps 38
Figure 5.4 Validation 4 with network B & 4 background apps 39
Figure 6.1 Time Taken to connect to WI-FI Network 41
Figure 6.2 Time Taken to Open the Application 42
Figure 6.3 Comparison between Time taken to connect to network and open the application 43
Figure 6.4 Difference between the test cases 44
Figure 6.5 Comparing both the test cases of Validation 1 48
Figure 6.6 Comparing both the test cases of Validation 2 51
Figure 6.7 Comparing both the test cases of Validation 3 54
Figure 6.8 Comparing both the test cases of Validation 4 57
Figure 8.1 Conclusion for Performance Testing 63

LIST OF TABLES

Table 4.1 Information for Network and Mobile App 23
Table 4.2 Information for different services used 25
Table 4.3 Information for different services used 28
Table 5.1 Test cases for Manual Testing 32
Table 5.2 Test cases for Automated Testing 33
Table 5.3 Test cases for Performance Testing 34
Table 5.4 Validations for Performance Testing 35
Table 6.1 Results for Manual Testing 40
Table 6.2 Average, Median and Standard Deviation for Connecting to WI-FI Network 41
Table 6.3 Average, Median and Standard Deviation to Open the Application 42
Table 6.4 Time Difference between test cases (in seconds) 44
Table 6.5 Average of View Result Tree of Login function for Validation 1 46
Table 6.6 Average of View Result Tree of Run Application for Validation 1 47
Table 6.7 Average of View Result Tree of Login function for Validation 2 49
Table 6.8 Average of View Result Tree of Run Application for Validation 2 50
Table 6.9 Average of View Result Tree of Login function for Validation 3 52
Table 6.10 Average of View Result Tree of Run Application for Validation 3 53
Table 6.11 Average of View Result Tree of Login function for Validation 4 55
Table 6.12 Average of View Result Tree of Run Application for Validation 4 56
Table 6.13 Comparison of failed samples (error %) for all validations 58

Chapter 1

INTRODUCTION

In today's internet life, mobile devices play an important role. Users can install applications which are designed to these devices. Mobile applications are used for different purposes i.e. playing games, sharing knowledge etc. An upgrading diverse interactive mobile application helps us in different domains of our daily life. When analysing the Quality of Experience (QoE) [1] relevant to mobile applications, the time required to display the data is a significant aspect. If the response time is longer than expected, the user will not be satisfied with the mobile application, even if it has interesting features. The QoE is defined as the degree of delight or annoyance of a customer's or user's experience of an application. It results from the fulfilment of user expectations with respect to the utility of the application. As a measure of end-to-end performance at the service level from the user's perspective, QoE is an important metric for designing the systems and to evaluate the performance of an application. Because due to the decrease in the performance of any application may affect the user's experience. So, when designing the systems, the expected QoE output is taken into consideration. Usage of QoE can improve the quality of network performance [2], improve the response times for authentication [3]. Web applications are also the crucial part in today's world. It is a client and server computer program which runs in browser. Usage of these web applications helps to learn anything in the world and can gain specific information about that application.

In this thesis, we test the performance of an iPad application. It is a company's application (PLANETJAKTEN). There is a delay in provisioning the application. But the information provided by them is, the application might be a traffic application or an edtech application. To gain knowledge and experience in testing of various platform applications and to present our previous experience in manual and automated testing, we performed such testing methods. BlekingeTrafiken is a mobile traffic application and Edmodo is a web edtech application. These are the main reasons for choosing such testing methods for the particular applications.

1.1 MANUAL TESTING

Manual Testing [4] is the process of using the functions of an application as an end user. Understanding the requirements of an application and writing the test cases are the crucial part of the manual testing. Manual Testing helps the researchers or testers to find the bugs of any application.

Usability testing [5] is a type of manual testing helps to visualize something by testing with real users. In this case, users are asked to complete tasks, observed by the researcher, to encounter problems they are facing. One of the mobile applications used for manual testing is BlekingeTrafiken. It is a public transport application which helps the users or customers to search for their travel.

1.2 AUTOMATED TESTING

Automated Testing [6] is the process of automating an application using automated tools. Using test scripts, all the process goes automatically without manual effort. Automated testing is also used to test the application with regards to load [7], performance and stress testing. Stress testing [8] is a type of software testing which determines the stability of software by testing beyond the limits. With the help of automation tools, we can write and execute the test cases. These days every application is being automated because of the increase in several automated tools and frameworks. Using test automation tools, it is possible to record the test suite and re-use it when required. Automated Testing helps to reduce the test cases, improves the overall speed of test execution. For any application (both mobile and web), automated testing helps to find the bugs flexibly, increases the efficiency, provides more accuracy, frequent execution, supports regression testing and lights out of execution. Lights out of execution mean even if the researcher is not present, testing goes automatically. It is also used to test the application from a different point of views. For load testing, performance testing and stress testing, automated testing helps to increase the process.

In this thesis, we used Selenium [9] as an automation testing tool for the web application. Selenium is a suite of software tools to automate web browsers. Selenium helps to customize the code for improvement in code functionality. The generation of test scripts to validate functionality is easy with selenium. It also supports dedicative framework which helps in writing test scripts. The implementation of selenium is easy because of its user-friendly interface. These are the main reasons of choosing selenium as an automated test tool. We may choose an alternative tool. But they are not open source and does not support all the programming languages. The brief description of Selenium is explained in the Background section (2).

In this thesis, we used EDMODO [10] application for our automated testing. EDMODO is a web application which acts as an online classroom. The purpose of choosing this EdTech application is it’s easy to learn and use. It provides an innovative (refines the ideas from social network) way of teaching for classroom. It provides the digital learning on multiple platforms and parents can actively check the status of their children’s learning. Using this application, we can log in as a student or as a teacher. We can do our assignments, record classes etc. This web application is completely related to the offline classroom.

1.3 PERFORMANCE TESTING

Usually, we tend to follow the trends in testing and forget to pay importance for verifying whether the software or application needs to be expected or required performance. Unfortunately, we figure out the performance issue after the post delivery of that application or software. So, performance testing [11] is to be done to detect whether the application performs well under extreme load conditions (increase in no of users etc). While doing performance testing, it mainly focuses on Speed, Stability and Scalability. Because the results matter a lot with application's functionality and features. We had chosen the end user perspective rather than the network perspective. Because the performance decreases at the end user side (by increasing the users) and not with the change in the network. To clarify that, we done some validations by changing the networks too.

Here we are testing the performance of network-based application but not with the network performance. The measurements of network performance are packet loss and jitters.

For some mobile or web applications, when there is an increase in several users, then automatically we can observe the decrease in performance of that application. To overcome those issues, we must do performance testing. The decrease in performance may occur due to bottlenecks [12] issues like CPU utilization, memory utilization or network utilization.

When we sent any request to the server then depending upon the service, if we received the response longer than expected then performance issues (failed requests) may occur.

For some applications, after sending the request, we can't get the response then performance issues may occur. Also, for some of the applications, on increasing the number of users there will be a delay in response times.

In this thesis, we had done performance testing of an iPad application [13]. For performance testing, Apache JMeter [14] is the tool we used. It is an open source and can record all the tests. It can also report the bugs. Finally, it can document the entire process of testing. Apache JMeter helps to analyse the overall server performance under heavy load. The main advantage of JMeter is it can test both static and dynamic resources. It provides the performance results in different ways (graphical, html, tabular etc). These are the main reasons of using Apache JMeter for performance testing tool. We may choose other performance tools. But they do not generate results in various formats (as JMeter does). The brief description of Apache JMeter is explained in the Background section (2).

1.4 PROBLEM STATEMENT

For mobile and web applications, the performance (network) issues affecting QoE is a big challenge. One of the problems is under heavy load (maximum number of users), the application may misbehave. The main challenge is to test the performance of the application, analyse the results and to validate those results under different conditions. For this process, we are using a performance testing tool (Apache JMeter) to identify the results and also monitoring tool (Nagios) [15] to track each and every part of the application which

is under test. The main aim is to measure the performance of the application under various networks [16] and to propose a methodology for performance testing.

1.5 RESEARCH QUESTIONS

1. What are the commonly perceived problems of a mobile application from performing Manual Testing?

2. What are the commonly perceived problems of an Edmodo application seen from performing Automated Testing?

3. What are the strategies that are required to prevent an application from slowing down when many users were using the application at the same time?

1.6 METHODOLOGY

In this thesis, we used three different types of testing. They are Manual Testing of mobile application (BlekingeTrafiken), Automated Testing of a web application (Edmodo) and Performance Testing of an iPad application (PLANETJAKTEN). Any application may be tested manually or automated, then decrease in performance can be observed at the user side for some applications. So, we had included these 3 different types of testing.

Firstly, we will test manually the mobile application by increasing the number of users and will note down the time measurements of corresponding test cases. We will calculate the median, standard deviation, average of those measurements and compare them.

Next, we will perform automated testing of an Edmodo application and calculate the time taken in between the test cases and compare them.

Next, we will test the performance of a PLANETJAKTEN application under various situations (changing networks, including background applications) and test cases. Finally, we will validate all those results based on the test cases.

1.7 THESIS OUTLINE

The overall thesis is divided as below:

Chapter 1: This chapter explains briefly about **Introduction** for the thesis. It also includes research questions and methodology.

Chapter 2: This chapter explains about **Background** work for the thesis. It provides a brief description of the background literature work of various tools included in the thesis.

Chapter 3: This chapter provides the information of **Related Work** of the thesis. It also explains about various types of testing and applications used in the thesis.

Chapter 4: This chapter explains the process of **Methodology** for Manual Testing of mobile application and Automated Testing of a web application and performance testing of an iPad Application. It also analyses the results of all the applications under test. The configurations between server and client, different test cases used, various inputs used, and analysis are discussed.

Chapter 5: This chapter visualizes the **Validations and experimental parameters**. It also explains about the test cases used in the thesis.

Chapter 6: This chapter explains about the **Results** for the Manual testing, Automated Testing and Performance Testing under various network parameters.

Chapter 7: This chapter provides the **Research Questions & Answers**

Chapter 8: This chapter provides the **Conclusion and future work** .

Chapter 9: This chapter provides the **References**.

Chapter 10: This chapter provides the **Appendix.**

Chapter 2

BACKGROUND

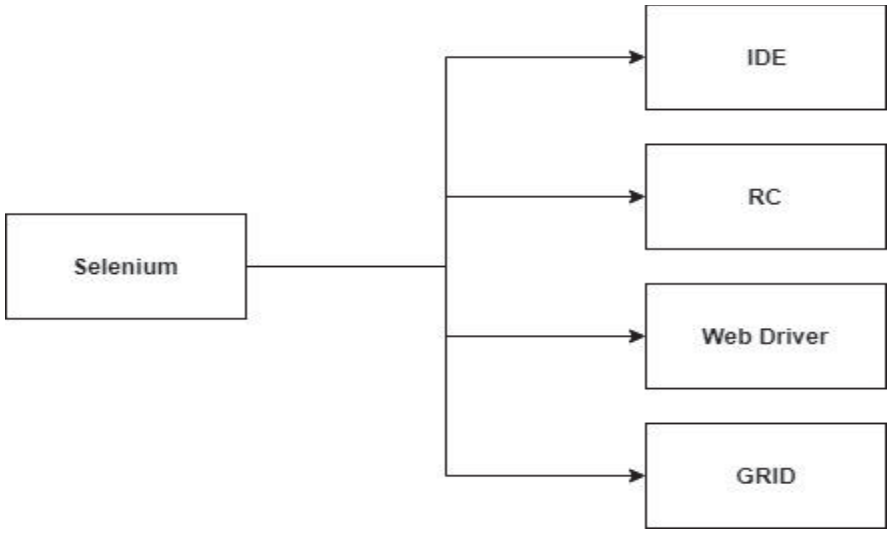
2.1 SELENIUM

It is open source and mainly used for functional testing and regression testing. Functional testing is a type of testing which verifies that each function of any application operates according to requirements. Regression Testing is a type of Software Testing used to update the existing features by re-testing. If any new code changes or functionalities occur, then the existing functionalities should not have any side effects. This type of testing helps to find bugs. It is developed by Jason Huggins in 2004.

The purpose of using selenium is it's an open source, supports most of the programming languages like Java, Python, C#, PHP, Ruby, Perl, JavaScript. It also supports different OS like Windows, Mac, Linux, IOS and Android. Selenium also supports different browsers like IE, Firefox, Chrome, Safari and Opera. Selenium supports parallel execution. The main drawback of Selenium is it does not support windows-based applications and difficult to test Image based applications.

Selenium has four components which are shown in the fig 2.1. They are as follows

Figure 2.1 Components of Selenium

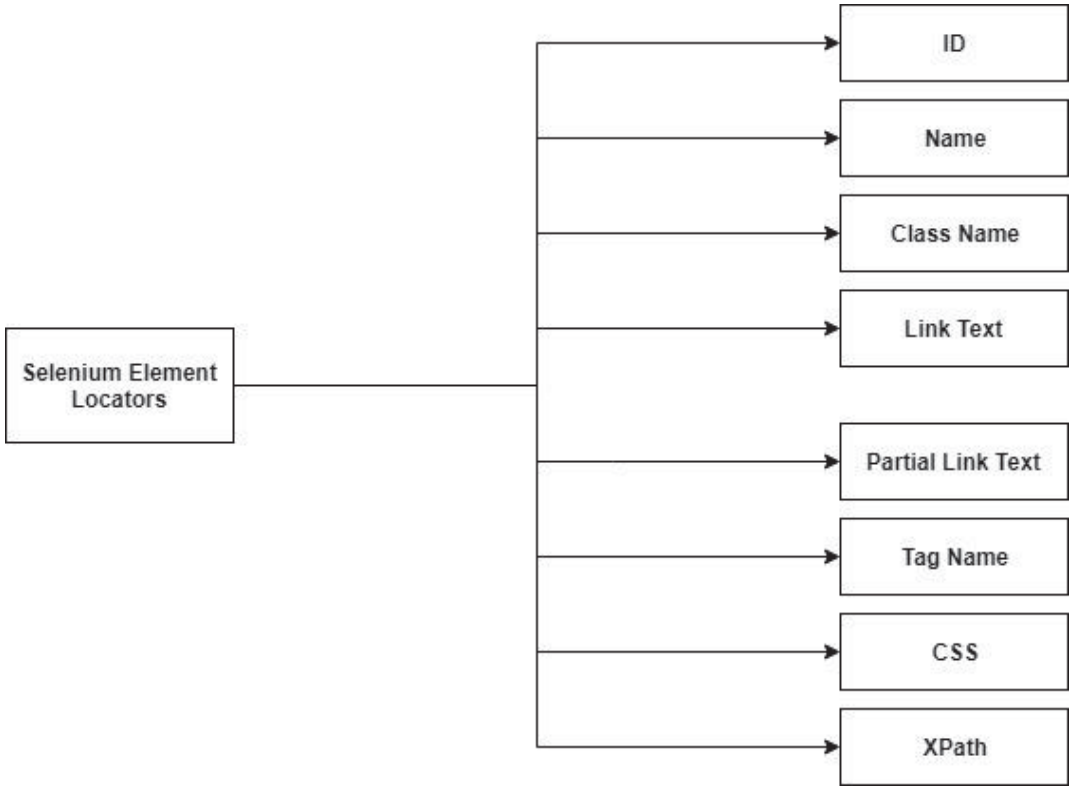


Selenium RC (Remote Control) [17] is used to test the automated web GUI tests in a programming language like JavaScript against any HTTP website. It comes in two parts. A server which automatically kills browsers and acts as an HTTP proxy for web requests. Also, client libraries for our favourite language. But it does not support for HTML Unit Browser. HTML Unit browser is a headless web browser in Java. It provides access to details for received

web pages. The architecture of Selenium RC is a bit complex. RC generates detailed test reports compared to the web driver.

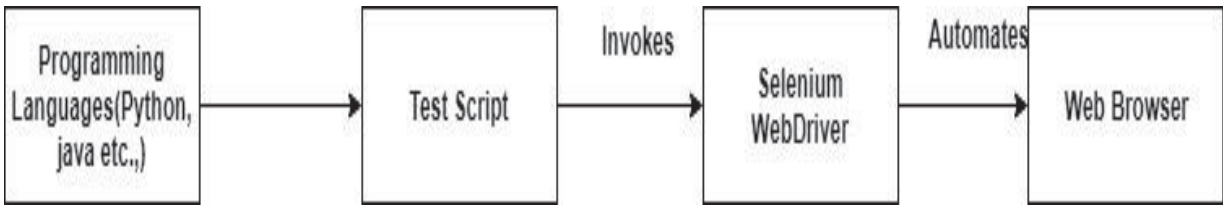
Selenium Web driver [18] is designed to be more understandable programming interface and has a simple architecture. Web driver is faster because it interacts directly with browser and user. Also, it supports for HTML Unit browser. In selenium web driver, there are a total of 8 Element locators (Object locators or WebDriver methods).

Figure 2.2 Element locators in Selenium



While automating, these Element locators which are shown in fig 2.2, helps to find the elements present on the web page. Test cases are created and executed using these Element locators. We will write our entire test script using an IDE (Eclipse) in web driver. With the help of web driver, we can test on the local machine. Using web driver alone is impossible to test on the remote machines. To overcome this issue, we must use both the RC Server and web driver. While testing on a remote machine, commands from web driver go to Selenium RC server which is then interpreted on the remote machine. Web driver [19] has an API which is simpler than RC's API. Web driver also supports Batch testing, Cross browser testing and data-driven testing.

Figure 2.3 Architecture of Selenium web driver



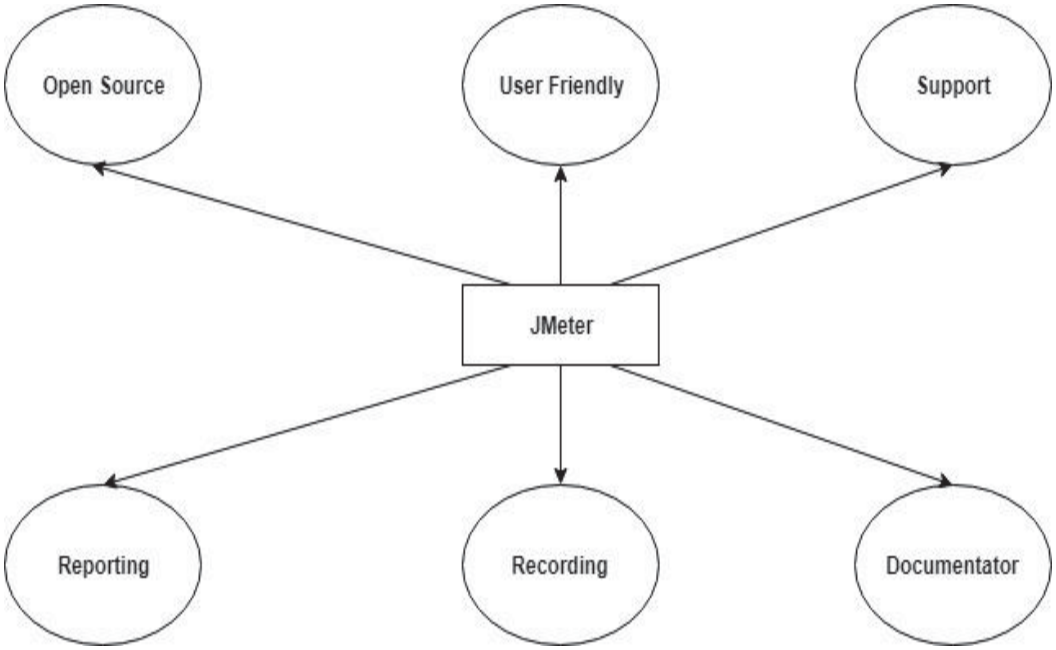
The above figure 2.3 explains about the working of Selenium web driver. Initially, a test script has been written on any language. After executing the test script, the web driver is invoked from test script and then automates to the web browser.

Selenium Grid is one of the components which helps us to run multiple test cases in parallel. If we want to run our test in multiple browsers on various operating systems parallel, then the grid is the best option. It acts as a time saver.

2.2 APACHE JMETER

Apache JMeter is a tool used to test the performance of mobile and web applications. It is used to test the performance of both static and dynamic resources. Using JMeter, we can test any type of applications/ server. Some of them are Web (HTTP/HTTPS), TCP, Java objects etc. The main reason for using Apache JMeter is it can calculate measurements [20] of each and every part of the application. In order to increase the performance of any application, we need to calculate the measurements under different situations (varying networks, an increasing number of users, varying storage of the device etc). Based on the validation results, we need to move forward (changing source code etc). As there are different methods for configuration to iPad, we used HTTP Proxy method. A proxy server is a server (system or application) used for sending requests from clients. Some of the measurements used in this thesis are Load time, connect time, latency, size in bytes, sent bytes etc. Apache JMeter helps to view the results in tabular form, graphical view etc. With the help of JMeter, it is possible to test maximum of 10000 users at the same time. The limitation of Apache JMeter is it consumes lot of memory under heavy load. It does not suit to test desktop applications.

Figure 2.4 Architecture of Apache JMeter



2.3 PLANETJAKTEN

PLANETJAKTEN is a gaming application [21] used to learning mathematics for school children. The release type of this application is BETA. The version used for performance testing is v0.1.4. This is the application which is accessible only through iPad. At the end of the session, the users will be able to feed the knowledge sample with the help of stars that collected during the training. A training session is thus what is meant for the students to run during a lesson. Its length can be set in minutes.

If any researcher is dealing with performance testing, then it's better to use a monitoring tool. Because bottlenecks are some of the reasons for decreasing the performance of an application or software. Bottlenecks are nothing, but the errors occurred on devices and network side (CPU utilization, Disk space, CPU memory etc). Bottlenecks can be decreased by updating, reconfiguring and replacing the devices. These can be easily tracked by the monitoring tool.

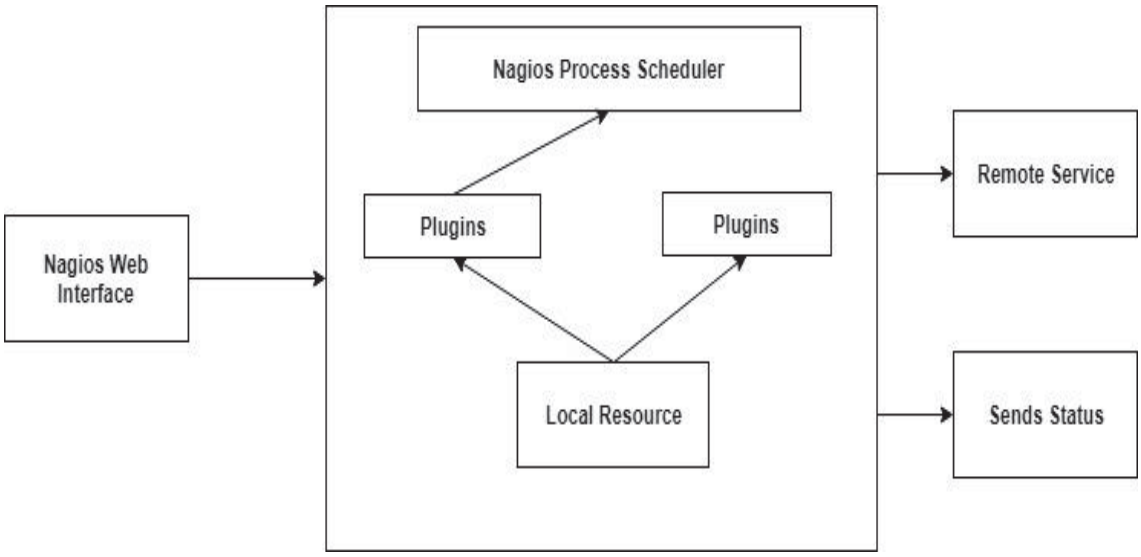
2.4 NAGIOS

In this thesis, we used Nagios as the monitoring tool for tracking the application. The purpose of using Nagios is it's an open source and mainly focuses on monitoring systems and networks. The main advantage of Nagios is it can give alerts to our systems if any damage (failure of components, low disk space etc) occurs. Nagios runs mainly with plugins [22]. if we want to monitor any OS, we need a corresponding plugin. For monitoring windows machine (in which proxy server is present), used Nagios XI and for monitoring iPad, used MobiosPush Plugin.

With the help of these plugins, we can monitor our systems and can analyse the performance issues.

Nagios is a powerful monitoring tool which helps to enable organisations to resolve IT infrastructure problems. Compared to remaining monitoring tools, nagios can detect applications or services with much faster, reliable. These are the main reasons for choosing nagios as a monitoring tool. We may choose other monitoring tools. But Nagios send alerts of any damage occurs in the service or system.

Figure 2.5 Architecture of Nagios



Architecture of Nagios is illustrated in fig 2.5. It completely works on plugins. Initially the web interface of Nagios is connected by entering the login credentials. Nagios scheduler is used to schedule and manage the plugins. After the configuration of plugins, the status can be seen in the dashboard.

Chapter 3

RELATED WORK

In this section, the related work of the research has been provided.

Initially, references [1], [2], [3] explains briefly about the overview of Quality of Experience and the work which is related to the networking, used in this thesis. These researches also explain about the mobile applications and network performances. For performance testing, we had validated the performance of the iPad application under various networks which is provided in [16].

References [4], [5] provide an overview of manual testing and usability testing of real-time applications. Initially, we had performed Manual Testing with the inclusion of Usability testing. For some applications, by increasing the no. of users, the time is taken for connecting to the network decreases. The Standard Deviation and Average also increases.

In today's life, automated testing plays an important role because of the increase in testing types and tools. In the references [6] explains briefly about Automated Testing and Stress Testing. For testing any type of application, test cases play an important role. Based on the test cases, we can find the bugs in the application. Automated Testing helps a lot because of the increase in test cases.

In [23], the test script for automation testing has been explained. In this thesis, using selenium, we had written the automated test script in Java. Also, all the functions present in the test script has been clearly explained.

References [7], [8] provides a complete overview of performance testing and load testing. While performing the testing for any application, load testing helps to understand the usage of maximum workload. For any application, load testing discovers the bottleneck issues.

References [9], [14], [15], [18], [22] presents all the tools used in the thesis. As dealing with performance testing and automated testing, usage of all these tools has a huge impact. With these tools, the comparison for various test cases has been taken. The usage of Selenium provides the time comparison study under different test cases. JMeter tool helps for measuring the performance of iPad application. According to the test cases, the actual bug has been calculated. For performance testing, monitoring the devices is the key role. Using Nagios tool and plugins, all the devices under test has been monitored.

In reference [12], [19], [24], [25], [26] provides complete information about the metrics and analysis included in this thesis. Any application's maintainability and reliability help to understand the analysis. In this thesis, we had performed an analysis of iPad application and web application. So, these software metrics provides the whole data of the application.

References [10], [13], [21] explains about the working of all types of applications used in this thesis. Background working of Edmodo application is provided. Similar gaming application used for children to learn mathematics has been provided.

In reference [27], [28] explains the complete overview of DevOps lifecycle. Nowadays every organisation is shifting towards DevOps. Because when any application is in the developing stage, immediately it can send to testing phase. So, all the errors can be fixed in the beginning stage of the application development. To increase the speed, scalability and performance of any application, the DevOps approach helps a lot. As continuous integration, continuous delivery and continuous deployment are a crucial part in DevOps, the performance of any type of application plays an important role.

References [29], [30] provides overall information about servers. As Nagios is used for monitoring the performance, a server should be needed to visualize the result. All the results (bottlenecks) can be displayed by those servers.

Chapter 4

METHODOLOGY

4.1 INSTALLATIONS

This section shows the installation process for tools, application, used in the research. We can visualize them below.

1. Initially we need to install Java. Because to install selenium or eclipse, java should be present. So, we installed java repository and Oracle Java with OpenJDK environment. The version for Java used is 8.0.
2. Next, we installed eclipse. Because for selenium web driver there should be an IDE. So, we installed eclipse neon from eclipse environment.
3. After installing java and eclipse, Selenium is installed from the official selenium website. Along with selenium, we had also installed selenium standalone jar file. This is a separate file which contains separate libraries which should be included in the eclipse for automation testing.

Figure 4.1 Installation of Selenium

Selenium 3.0 is installed. The entire process of Selenium is explained in the fig 4.1

4. To install Apache JMeter, java 8+ is the requirement. As we already installed java, we can install Apache JMeter from official website. A zip file will be downloaded with a lot of binaries. After unzip, we can see the Apache JMeter root CA certificate file. We need to install or update the file. It is valid only for 7 days. The version used for Apache JMeter is v4.1
5. This is an iPad Application which can't be installed from Apple store. Go to the website i.e. www. hockey app .net. This is the link address where the iPad application can be installed. As we need to test the application, company had provided the login credentials. After login, install the application. Once the application is installed, go to our iPad settings. Under profiles section, a trust certificate naming "Karolinska Institutet" is generated. After accepting the certificate, our application is ready to use.
6. We installed Nagios Core in ubuntu. Initially, we downloaded and extracted the Nagios Core in ubuntu. Then started compiling Nagios. We had configured Apache server [29]. We installed Nagios Core Plugin to windows (Apache JMeter is present in Windows machine) and MobiosPush Plugin to iPad (Planetjakten application is present in iPad). Configuration for both the plugins has been done. It looks like

'/usr/local/nagios/etc/nagios.cfg'. Then, we started the Apache server and Nagios. Then, we tested the Nagios Server [30] by entering the login credentials. We can access to the Nagios Dashboard. Then we had recorded all the results and done some validations.

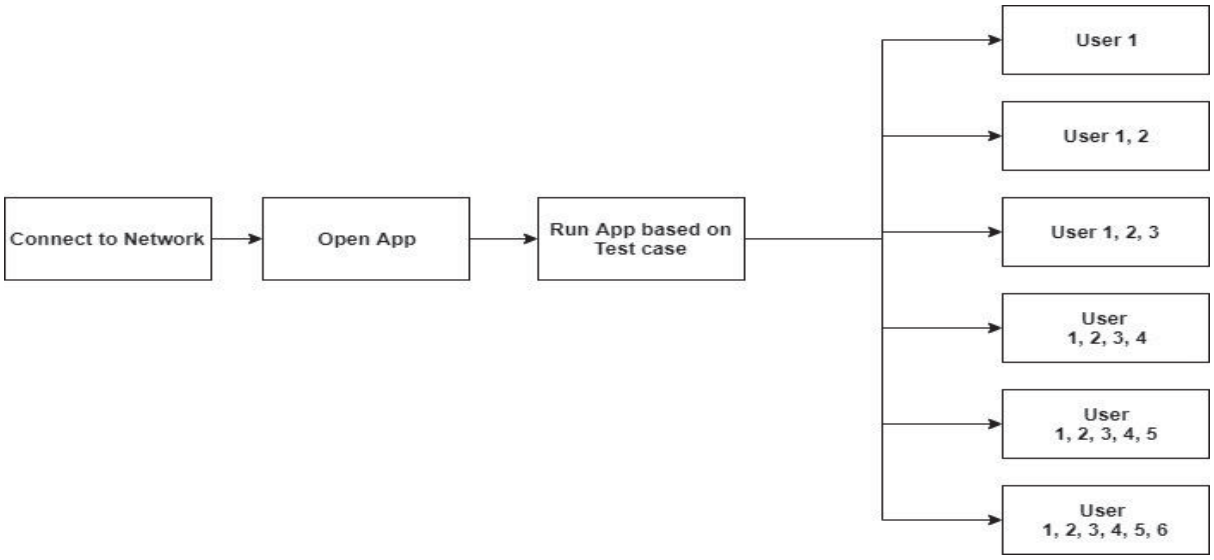
4.2 EXPERIMENTAL SET UP - ARCHITECTURES

This section explains the information about different architectures involved in our thesis.

4.2.1 MANUAL TESTING

The architecture for manual testing of a mobile application is shown in fig 4.2. The process of this experiment goes on based on test cases.

Figure 4.2 Experimental Setup for manual testing



The above diagram is experimental set up for manual testing. Initially all the users will connect to same network. Then users will open the application at the same time. Now users will run the application based on test cases. This process continuous by increasing the users up to 6 users. Below is the table regarding the information of services.

Table 4.1 Information for Network and Mobile App

No. of Services	Service	Name/ IP
1	Network	193.11.185.151
2	Mobile Application	Blekinge Trafiken

The above table describes about the services used for manual testing. The IP address of the network which connected is 193.11.185.151. The mobile application used is BlekingeTrafiken.

For manual testing, the mobile application used is BLEKINGE TRAFIKEN. It is completely related to usability testing. Initially, all the users are connected to the same network. The application is installed on all the devices in which the users are testing. The testing goes on by increasing the number of users based on the test cases used.

1. The researcher will switch on WI-FI. User 1 will run the application based on test cases. The researcher will start the stopwatch and note down the time required for every test case. The researcher will switch OFF WI-FI.

2. For the next step, the researcher will switch on WI-FI. User 1, User 2 will run the application at the same time. The researcher will note down the measurements for both the users and switch OFF the WI-FI.

3. Similarly, the testing continues by increasing the users from User 3 to User 6. The researcher or tester will note down all the measurements for all the users. With this type of testing, users and researcher can observe the exact problems.

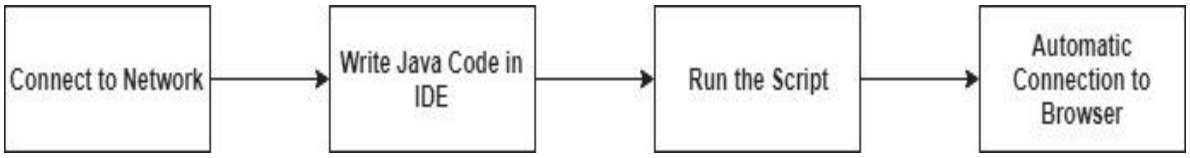
4. Once the measurements are taken for all the users, the researcher will calculate the Average, median and standard deviation.

5. Based on the final output of Average, median and standard deviation, graphs have been drawn.

4.2.2 AUTOMATED TESTING

The architecture for automated testing of a web application is shown below. We had tested each part of the application based on the script written in java and test cases (mentioned in section 5.1.2). We had also included the time stamps. Initially the researcher will connect to the network and write the test script. We used java for automated test script. After executing the script, the browser gets connected and all the process goes automatically.

Figure 4.3 Experimental Setup for Automation testing



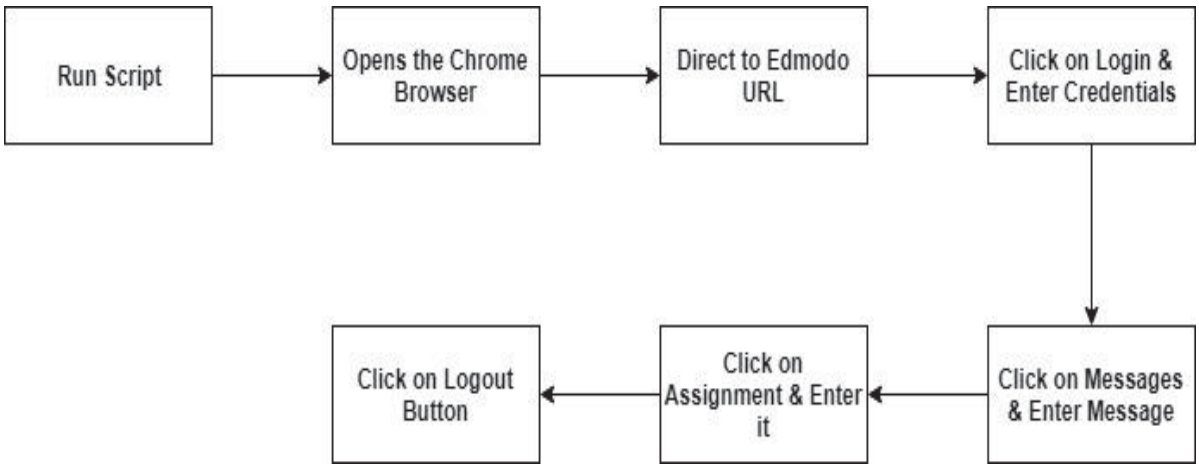
Below is the table regarding the information of services used for Automated Testing of web application. Java is the programming language used and the IP address of network which gets connected is 192-168.0.120. The automated tool used is Selenium 3.3. Firebug is a plugin used to find the elements is the page. It helps to automate the process. The web application used is Edmodo.

Table 4.2 Information for different services used

No. of Services	Service	Name/IP
1	Programming language	Java
2	Network	193.11.185.151
3	Automated Tool	Selenium 3.3
4	Plugin	Firebug
5	Web Application	Edmodo

In this section, it explains about the automated testing of a web application. The application used is EDMODO. Selenium is the automated tool used. The entire process goes on based on the script written in Java [23]. Eclipse is the IDE used to write the script for automated testing. We had also included Gregorian Calendar to see the time taken in between the test cases. The browser used is Chrome. As discussed in the introduction, there are 8 element locators. With the help of these element locators, we had done automation for Edmodo web application. To detect these elements, present in the web page, Firebug is the plugin used. Firebug is a plugin used to detect the elements on any web page.

Figure 4.4 Architecture of Automated Testing for Edmodo Application



1. Selenium WebDriver will get connected to the Chrome Driver (used to control the Chrome).
2. Now, WebDriver gets connected to our website (Edmodo) for testing. We used a 'GET' command to open the URL.

Syntax: driver.get("URL");

3. It automatically clicks on the Login Button using 'Find Element' command.

'Find Element' is a command used to find the elements present in the Edmodo web page.

The syntax to 'find element' is

WebElement elementName = driver. findElement(By.Locator("LocatorValue"));

Locator = element locators

Locator Value = URL

4. It automatically enters the login details using 'sendKeys' command.

'SendKeys' is a command used to send characters on the web page. The

Syntax for 'SendKeys' is driver. findElement (By.Locator("locator value")).sendKeys("characters");

5. We had created a class group (Teacher and 30 students). So, it automatically clicks on the messages and enters like "Did you submitted your homework?"
6. It automatically navigates back using navigation command.

'Navigation back' is a command used to navigate to the previous web page.

The syntax for 'Navigation back' is driver.navigate().back();

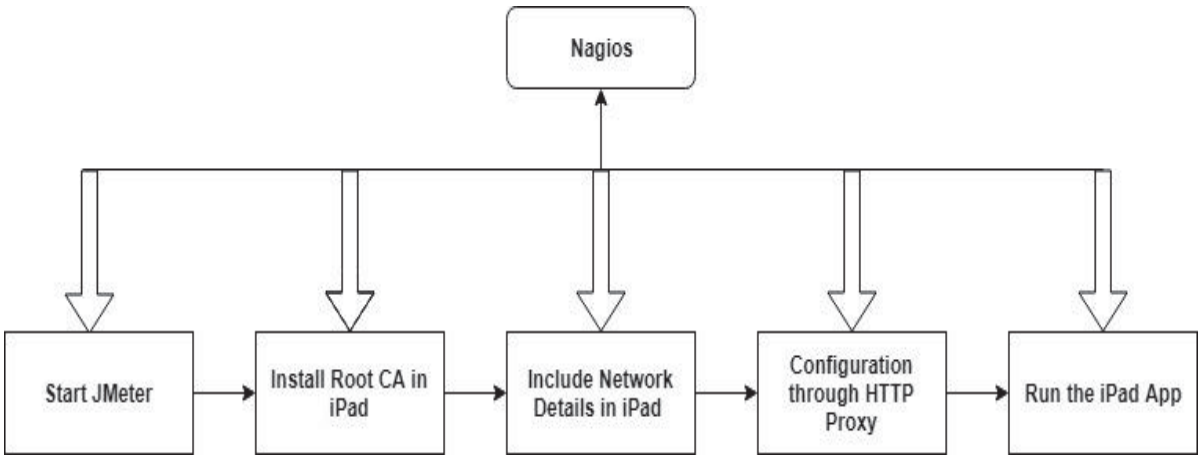
7. It automatically clicks on the Assignment button. The teacher will assign an assignment to the students and click on the log out button.
8. In the middle of the test cases, we had included time to know the time taken in between each test case.

In the entire process, if there are any bugs, we can see them in the output.

4.2.3 PERFORMANCE TESTING OF AN IPAD APPLICATION

The architecture for performance testing of an iPad is shown below. It explains the process of setting up the infrastructure to getting up the results. This process is performed when both the laptop and iPad are connected to the same network. The test cases for performance testing are explained clearly. For iPad application, to log in to the application and run the application are test cases which kept as constant. Some of the parameters included are Load Time, Connect Time, Standard Deviation, Confidence Intervals and Error % (no of failed samples). The entire performance testing has been monitored by nagios monitoring tool. Initially the JMeter should be started and install the root certificate on iPad. Then all the network details (IP address, port number given in JMeter) should be included in Proxy settings of iPad. Now the application should be started, and the measurements should be noted.

Figure 4.5 Experimental Setup for Performance testing



Below is the table regarding the information of services used for performance testing of an iPad application.

Table 4.3 Information for different services used

No. of Services	Service	Name/IP	Version
1	Performance Tool	Apache JMeter	4.0
2	IOS App	Planetjakten	0.1.4
3	Networks	193.11.185.151, 193.11.184.186	–
4	Monitoring Tool	Nagios	4.3.4
5	Plugins	Nagios XI, MobiosPush	–

We had done performance testing of an IPAD application for more than 4 minutes.

1. Initially, we need to start the Apache JMeter present in our system. The operating system used for this test is Windows.

2. We had created a Test Plan in the JMeter. Test Plan is a container which contains all the elements of the test.

3. Then we had added Thread Group element inside the Test Plan. Thread Group means users will be allowed to run or use this test.

4. As we must increase the no of users for this test, initially started with 1 user. Also, we had included Ramp-up [24] Period of 1 second in this test. Ramp-up period is how long the time should be taken before starting the next user (thread) chosen. Suppose, if there are 50 users and a 50-sec Ramp-up period, then the time taken before starting the next user would be 1 second (50 sec/50 users).

5. We had added Samplers to our test. Samplers are the type of requests [25] that JMeter should handle. We had used the HTTP Request for testing our iPad Application. In this section, we had included the port number 8080 to test our application.

6. We had also added Listeners to our test. Listeners are nothing but results/ reports which can be visible in various types. As JMeter provides several results, some of the Listeners included are Aggregate Graph, Assertion Results, Backend Listener, Graph results, Response Time Graph, View Results Table, View Results Tree.

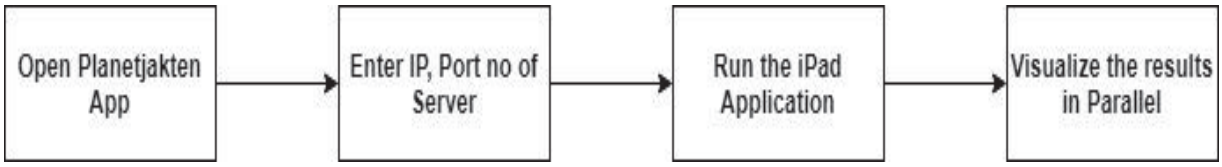
Once the test has been created, we started the JMeter present in the Windows.

Immediately, it will ask to install the root CA Certificate. The root certificate is a public key certificate which identifies the matched authority. It is a secure physical distribution. This certificate is not considered as valid unless it has been signed by a trusted CA.

JMeter will generate a root certificate which should be signed by the researcher and then it is installed in the iPad where the application is present.

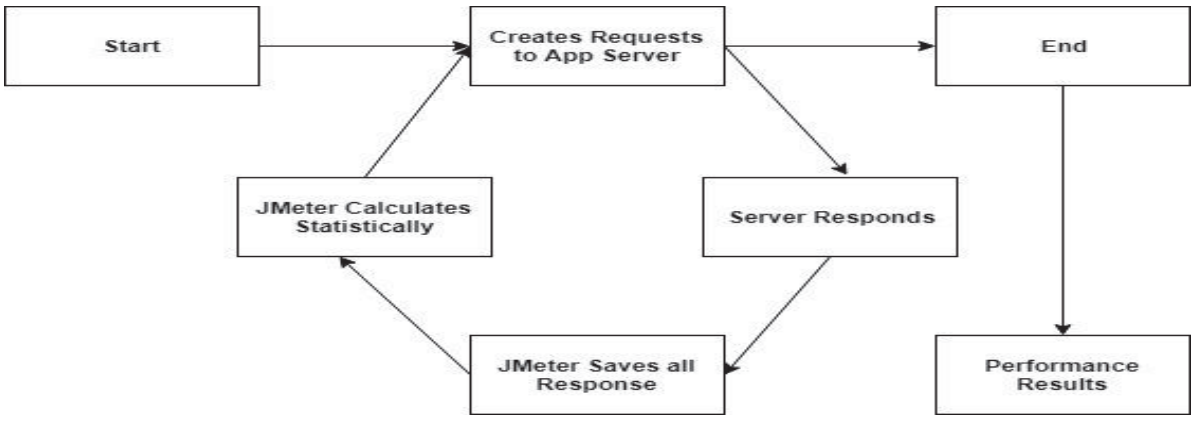
In the iPad settings (WI-FI/Proxy), we had manually given the IP address of the windows machine and port number (8080) which is present in the Apache JMeter tool. Both the windows machine and iPad should be connected to the same network.

Figure 4.6 Process for Root Configuration



We must start using the iPad Application. The request which we send from the mobile app will get connected to the JMeter. From JMeter, we will get the response of the application. The reason is whenever the root configuration and HTTP Proxy method are done, the iPad is controlled by JMeter. JMeter will capture all the actions done in the iPad app and records them. Whenever we get any response, they can be visible in the JMeter tool. After the completion of this test, we had increased the users and ramp-up period changed the networks and recorded all the results.

Figure 4.7 Architecture of JMeter for iPad application



The above diagram 4.7 shows the architecture of Apache JMeter for this iPad application. Initially JMeter will be started. After the configuration to iPad, it will create the requests to

the app server (while using the application). The server responds and JMeter will save all the responses. All the measurements are calculated statistically by JMeter. Now, if there is any other request, then the whole process goes accordingly as mentioned above. Once the process is completed, JMeter will display all the performance results.

Throughout the entire process of performance testing of this application, we had monitored every process of all the systems used. Monitoring is essential in every process. Especially when the performance of any application is being tested, monitoring plays a crucial role. As Nagios is the tool used, always keeps tracking the status of the system. It alerts when there is any problem occurs in the system. Some of them are Server Disk Space, file system status, status changes in the services etc.

Figure 4.8 Architecture of Nagios in our Project

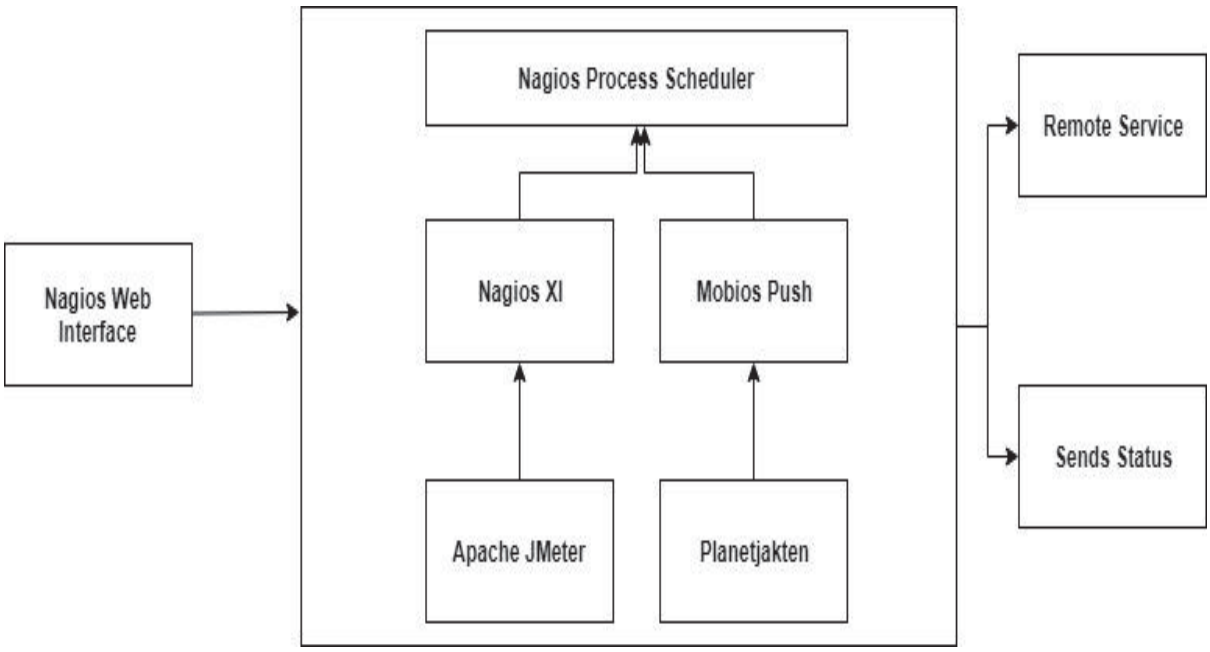
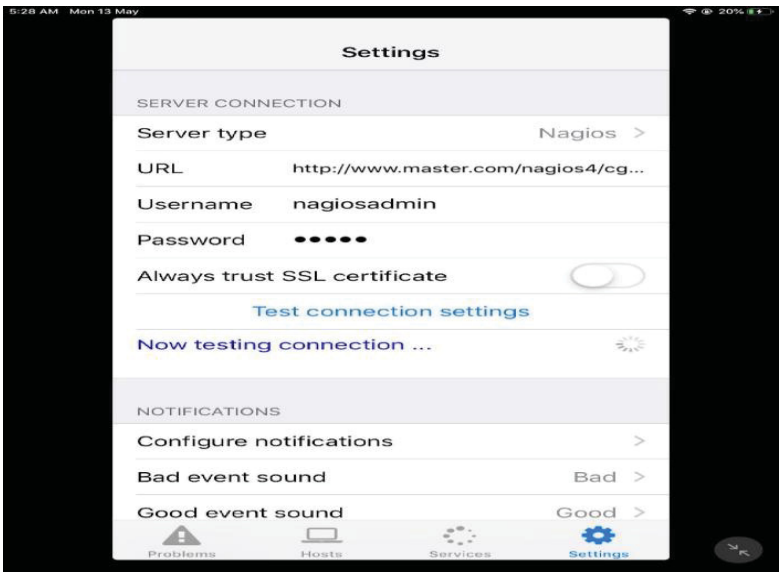
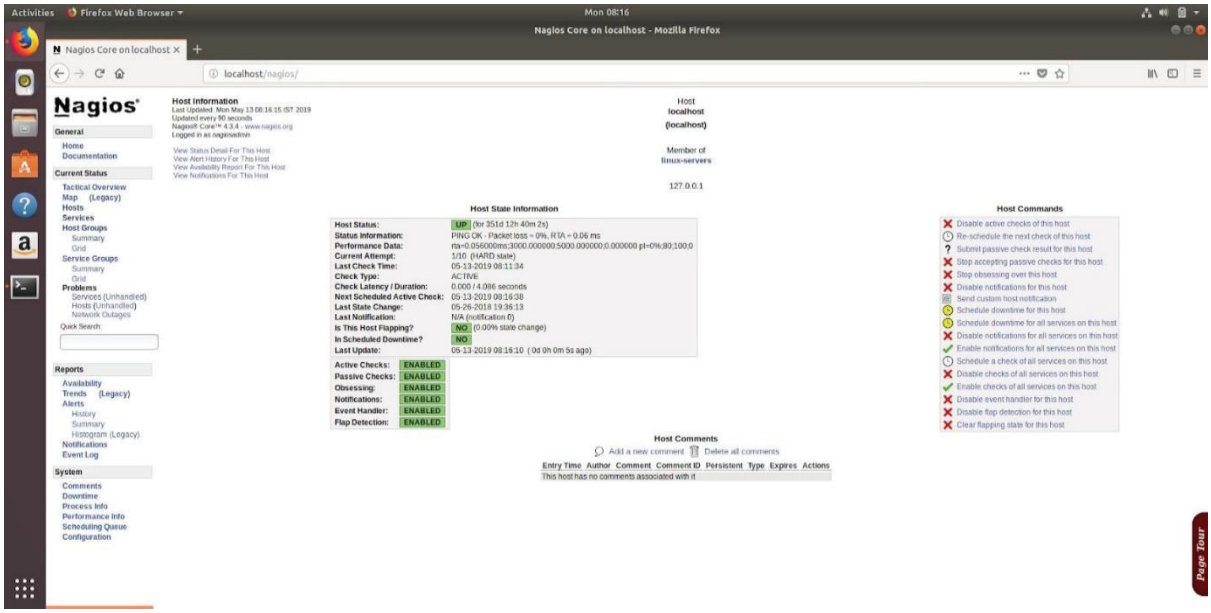


Figure 4.9 Configuration of MobiosPush



The above figure 4.9 explains about the configuration from MobiosPush plugin to Nagios. This configuration is set up in the iPad. The URL is the location of the nagios configuration file present in the Ubuntu. Those credentials are login details ton access to nagios.

Figure 4.10 Nagios Dashboard



The above figure 4.10 shows the dashboard of nagios after setting up the configuration. It shows about the hosts, host groups and services which nagios is monitoring. Nagios has 4 services. They are OK, Warning, Critical and Unknown. If the service works properly then it appears as “OK”, if the service does not work properly then it appears as “Warning”, if the service was not running or any damage occurs then it appears as “Critical” and invalid commands given in the service leads to “Unknown”.

Chapter 5

VALIDATIONS & EXPERIMENTAL PARAMETERS

This section presents the implementation of test cases for manual testing, automation testing and performance testing. It explains about the various validations used for performance testing.

5.1 TEST CASES

Initially we had gone through all the applications and understood the working of those applications. After that we understood the main steps of those applications and written the test cases.

5.1.1 Test Cases for Manual Testing of Mobile Application

Table 5.1 Test cases for Manual Testing

Test Case	Test Case Description	Test Steps	Test Data
1	Connect to WI-FI network	Click on WI-FI Button	WI-FI name: Eduroam
2	Open the Mobile Application	Click on the App	App Name: BlekingeTrafiken

The above table 5.1 explains in detail about the different test cases used for Manual Testing of mobile application. Initially the users will connect to the network by clicking on the WI-FI button on their mobile. The WI-FI used is Eduroam and IP address is 193.11.185.151.

Next the users will open the mobile application (BlekingeTrafiken) by clicking on the application. Based on these test cases, manual testing has been performed.

5.1.2 Test Cases for Automated Testing of a Web Application

Table 5.2 Test cases for Automated Testing

Test Case	Test Case Description	Test Steps	Test Data
1	Connect to Browser	Run the script written in Java	Browser Name: Chrome
2	Connect to URL	Automatically connects to URL	URL Name: Edmodo.com
3	Click on login button	Automatically clicks on login button	–
4	Enter the login credentials	Automatically	id:xxxxxxxxxx
		enters the login details	Password: xxxxxxx
5	Enter the message	Automatically enters the message	Message: Did you submitted your homework
6	Enter the Assignment in group	Automatically enters the message	Assignment: This is new Assignment
7	Click on logout button	Automatically clicks on logout button	–

Above table 5.2 explains about the test cases included for Automated Testing of a web application.

Initially the researcher will connect to the browser (Chrome) by executing the script written in java. It will automatically connect to the Edmodo web app (Edmodo.com) and clicks on the login button. Then it automatically enters the login credentials (user id, password) and enters the message as “Did you submitted your homework”. Then it automatically enters the Assignment as “This is new Assignment” and clicks on the logout button.

5.1.3 Test Cases for Performance Testing of iPad Application

Table 5.3 Test cases for Performance Testing

Test Case	Test Case Description	Test Steps	Test Data
1	Open the App on iPad and enter login credentials	Click on the App and login button	App Name: Planetjakten Password: xxxxxxx
2	Run the App	Click on various levels	–

The above table 5.3 explains about the different test cases included for Performance Testing of an iPad application. Initially the tester will open the application and enter the login credentials
(password) and run the application.

5.2 VALIDATION FOR PERFORMANCE TESTING

As we are dealing with performance testing, validation [26] is highly important. Because of validation, it is easy to find the bugs present in the application. As we are dealing with performance of the application, there might be an error in the networking side or heavy load on the device. By solving those errors, the performance of the application increases.
Some of the validations included in this experiment are

- Change in networks
- Change in background apps
- Change in test cases.
- Increase in users

For entire experiment the networks used are Network A and Network B. The name of the WI-FI Network A is Eduroam and name of the WI-FI Network B is Trossogatan.

- IP address of Network A = 193.11.185.151
- IP address of Network B = 192.168.0.82

We had included some background applications to test the performance of an iPad application. Usage of these background applications helps to find the performance when there is heavy load on the application. The 4 background applications used in this thesis for experiment are Google Chrome, Internet Explorer, Safari and Gmail. For each validation, the experiment is executed for 10 times. As the performance of the application decreases for every 5-6 frequent executions (mentioned by company).

Table 5.4 Validations for Performance Testing

Case	Validations	Test cases
1	Validation 1	Network A, no background applications, Login to the iPad application
2	Validation 1	Network A, no background applications, Run the iPad application
3	Validation 2	Network B, no background applications, Login to the iPad application
4	Validation 2	Network B, no background applications, Run the iPad application
5	Validation 3	Network A, 4 background applications, Login to the iPad application
6	Validation 3	Network A, 4 background applications, Run to the iPad application
7	Validation 4	Network B, 4 background applications, Login to the iPad application
8	Validation 4	Network B, 4 background applications, Run the iPad application

The above table 5.4 provides the detailed version of test cases included in the experiment. We are expecting that there might be an extra workload in the networks and device.

For Login function test case: Here experiment is executed until the login test case is passed. The no. of samples required to pass the test case is noted.

For Run the App: The prerequisite is to execute the experiment for 4 minutes (requested by the company). So, after the completion of 4 minutes, the no. of samples presented to run the application is noted.

5.2.1 VALIDATION 1

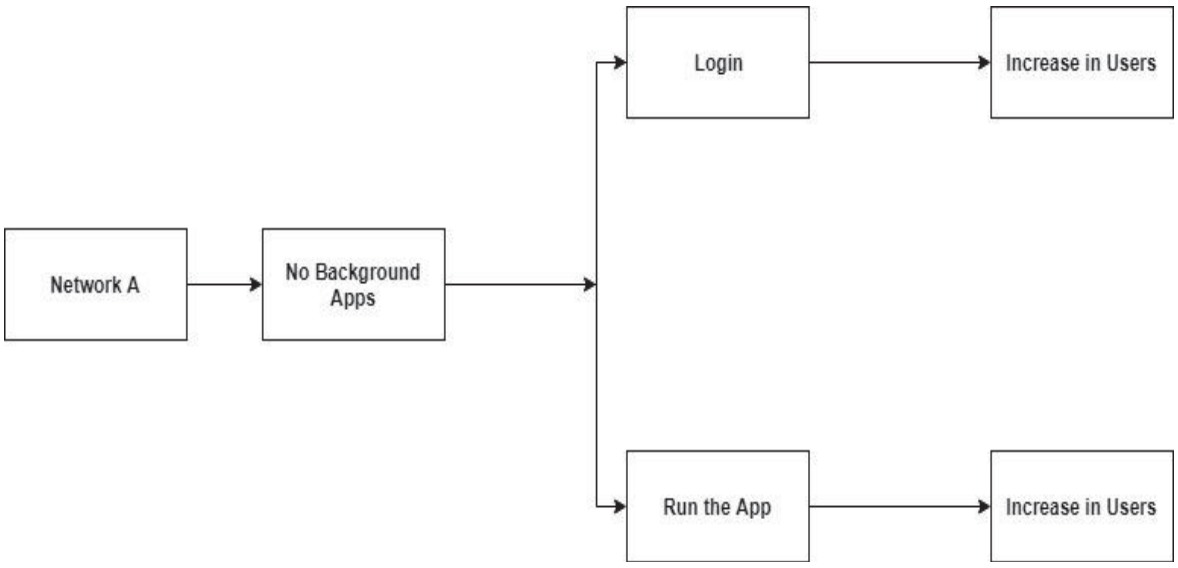
Test Case with network A, No background applications & Login

Initially, we understood the entire application and divided the test cases according to the working of the iPad application. The iPad application works mainly with Login function and playing the game. Initially, we had connected to network A. There are no background applications running on both the devices (windows machine and iPad). By starting the experiment with 1 user, the tester will click on the login button. After logging into the application, the readings are noted in the results section. Now, the application is restarted by increasing the users. In the same procedure, the readings are noted up to 10k users.

Test Case with network A, no background applications & Run the application

In this case, the tester or researcher is connected to network A. There are no background applications running on both the devices (windows machine and iPad). By starting the experiment with 1 user, the tester will login to the application and start using the application. While using the application up to 4 minutes, the entire readings are noted. Now, the tester will restart and use the application by increasing the users. In the same procedure, the readings are noted up to 10k users.

Figure 5.1 Validation 1 with network A & no background apps



The above figure 5.1 shows the following test cases for Validation 1. Initially, the user is connected to the specific network A and there are no background applications running on those devices. As per the first test case, the user will login to the application and then, the user will run the application after the completion of the first test case. In both the cases the no. of users will goes on increasing.

5.2.2 VALIDATION 2

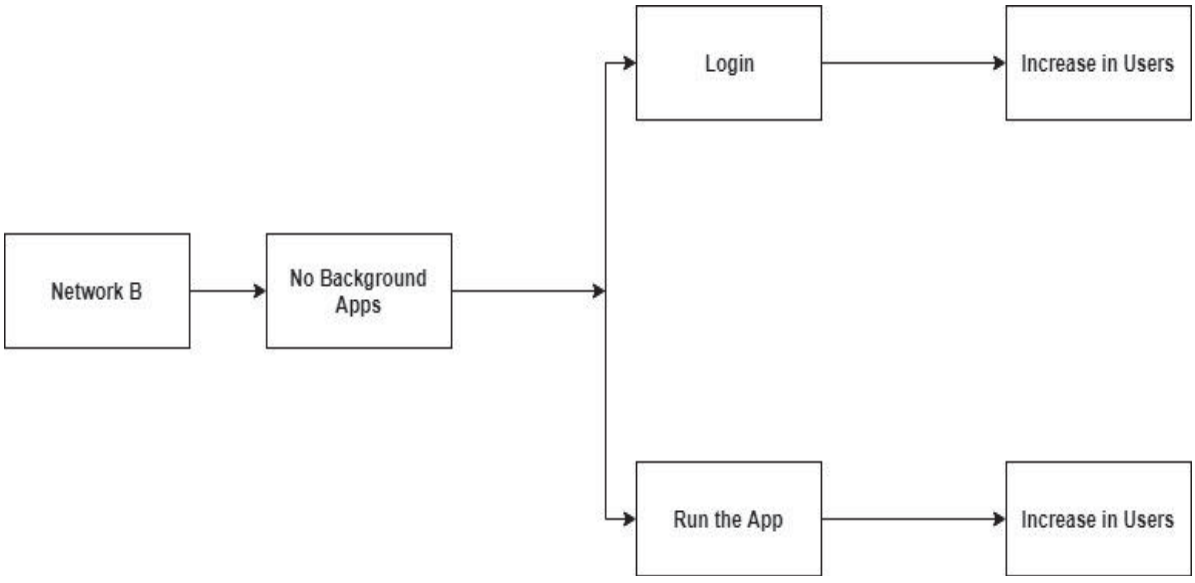
Test Case with network B, No background applications & Login

Initially, we had connected to network B. There are no background applications running on both the devices (windows machine and iPad). By starting the experiment with 1 user, the tester will click on the login button. After logging into the application, the readings are noted in the results section. Now, the application is restarted by increasing the users. In the same procedure, the readings are noted up to 10k users.

Test Case with network B, no background applications & Run the application

In this case, the tester or researcher is connected to network B. There are no background applications running on both the devices (windows machine and iPad). By starting the experiment with 1 user, the tester will login to the application and start using the application. While using the application up to 4 minutes, the entire readings are noted. Now, the tester will restart and use the application by increasing the users. In the same procedure, the readings are noted up to 10k users.

Figure 5.2 Validation 2 with network B & no background apps



The above figure 5.2 shows the following test cases for Validation 2. Initially, the user is connected to the specific network B and there are no background applications running on those devices. As per the first test case, the user will login to the application and then, the user will run the application after the completion of the first test case. In both the cases the no. of users will goes on increasing.

5.2.3 VALIDATION 3

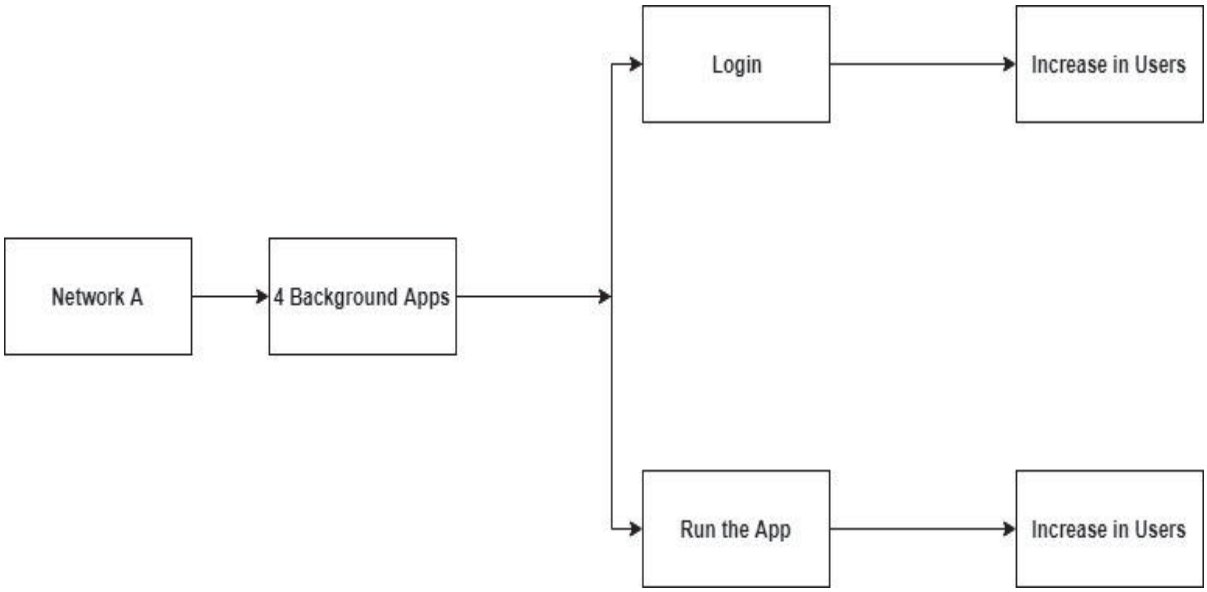
Test Case with network A, 4 background applications & Login

Initially, we had connected to network A. Each of the 2 background applications running on both the devices (windows machine and iPad). The purpose of using the background application is to increase the load on both the devices. By starting the experiment with 1 user, the tester will click on the login button. After logging into the application, the readings are noted in the results section. Now, the application is restarted by increasing the users. In the same procedure, the readings are noted up to 10k users.

Test Case with network A, 4 background applications & Run the application

In this case, the tester or researcher is connected to network A. Each of the 2 background applications running on both the devices (windows machine and iPad). The purpose of using the background application is to increase the load on both the devices. By starting the experiment with 1 user, the tester will login to the application and start using the application. While using the application up to 4 minutes, the entire readings are noted. Now, the tester will restart and use the application by increasing the users. In the same procedure, the readings are noted up to 10k users.

Figure 5.3 Validation 3 with network A & 4 background apps



The above figure 5.3 shows the following test cases for Validation 3. Initially, the user is connected to the specific network A and there are 4 background applications running on those devices. As per the first test case, the user will login to the application and then, the user will run the application after the completion of the first test case. In both the cases the no. of users will goes on increasing.

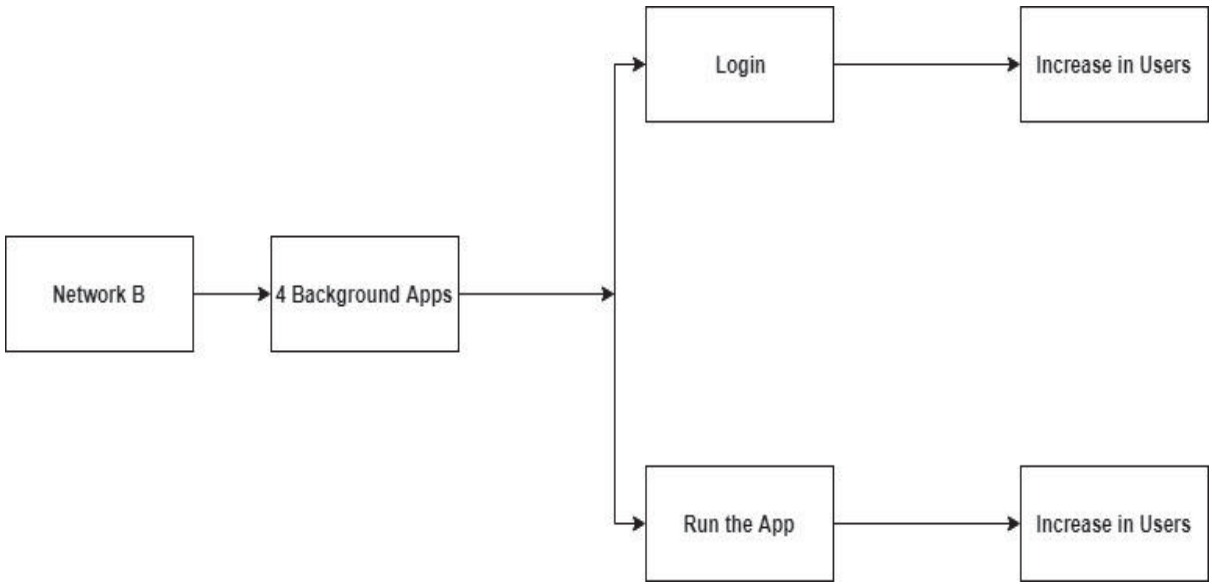
5.2.4 VALIDATION 4 Test Case with network B, 4 background applications & Login

Initially, we had connected to network B. Each of the 2 background applications running on both the devices (windows machine and iPad). The purpose of using the background application is to increase the load on both the devices. By starting the experiment with 1 user, the tester will click on the login button. After logging into the application, the readings are noted in the results section. Now, the application is restarted by increasing the users. In the same procedure, the readings are noted up to 10k users.

Test Case with network B, 4 background applications & Run the application

In this case, the tester or researcher is connected to network B. Each of the 2 background applications running on both the devices (windows machine and iPad). The purpose of using the background application is to increase the load on both the devices. By starting the experiment with 1 user, the tester will login to the application and start using the application. While using the application up to 4 minutes, the entire readings are noted. Now, the tester will restart and use the application by increasing the users. In the same procedure, the readings are noted up to 10k users.

Figure 5.4 Validation 4 with network B & 4 background apps



The above figure 5.4 shows the following test cases for Validation 4. Initially, the user is connected to the specific network B and there are 4 background applications running on those devices. As per the first test case, the user will login to the application and then, the user will run the application after the completion of the first test case. In both the cases the no. of users will goes on increasing.

Chapter 6

RESULTS & ANALYSIS

In this section, it explains about the results and analysis for all three types of testing included in this thesis.

6.1 MANUAL TESTING OF MOBILE APPLICATION

Comparing the time taken when connecting to WI-FI and opening an application by increasing no of users

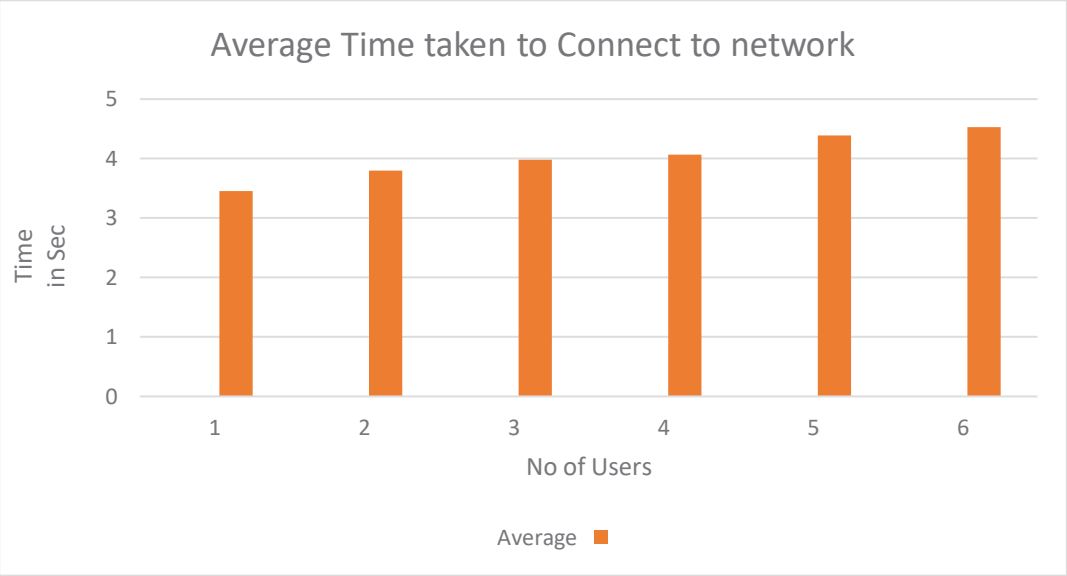
Table 6.1 Results for Manual Testing

No. of Users	Time Taken to connect to WI-FI (in Sec) (Xi) (X1….X6)	Time Taken to open an App (in Sec) (Yi) (Y1...Y6)
1	3.30	1.90
2	3.34, 3.96	1.96, 2.46
3	3.67, 4.14, 3.67	2.04,2.64,1.86
4	3.78, 4.14, 3.96, 3.78	2.54,4.39,1.86,1.86
5	4.0, 2.65, 4.08, 6.34, 4.04	2.04,2.04,2.04,2.64,2.04
6	4.57,4.88,3.78,7.26 2.90, 2.90	2.91,3.62,2.90,2.78, 1.94, 3.62

This above table 6.1 provides the comparison between time taken to connect to the network and opening the application. We had performed manual testing for up to 6 users. As it is manual testing, we had increased up to 1 user for every test. We had calculated the timings through stopwatch. Whenever the user clicks on the WI-FI button, the stopwatch starts. The time is stopped once the test is completed. The specific reason for testing up to 6 users is because of increase in time taken to connect to network and to open the application. When

the users go on increasing the time taken to execute those test cases increases. X1, X2, X3, X4, X5, X6 are the time taken while switching on WI-FI for users 1,2,3,4,5,6
Y1, Y2, Y3, Y4, Y5, Y6 are the time taken while opening the application for users 1,2,3,4,5,6

Figure 6.1 Average Time Taken to connect to WI-FI Network



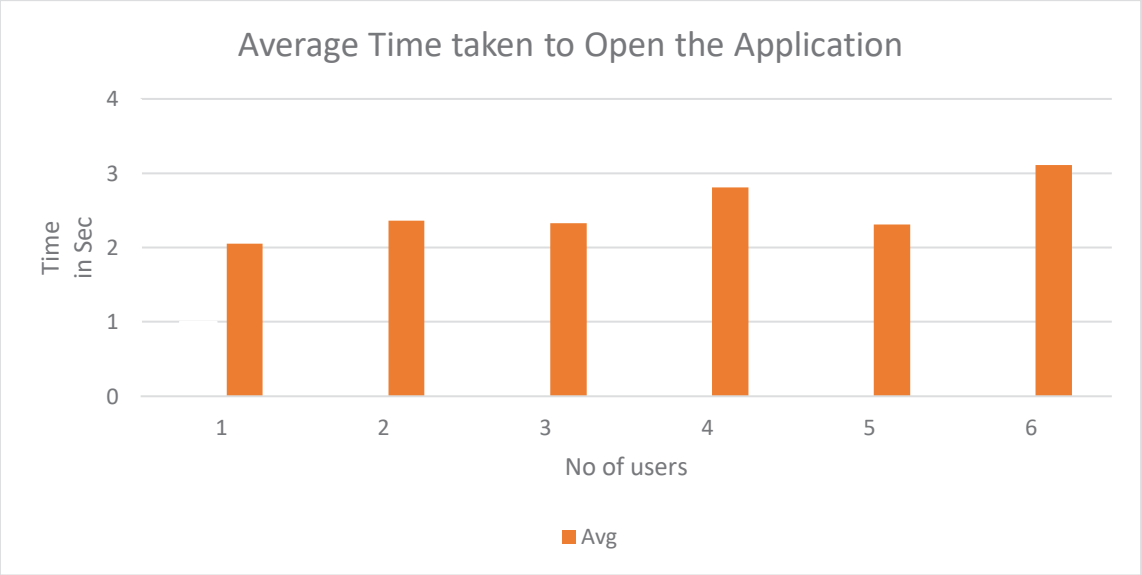
By referring to the graph shown in Fig 6.1, we can observe that by increasing the number of users, the time taken to connect to network also increases. Especially, for the last couple of values, there is much variation in graph.

Table 6.2 Average, Median and Standard Deviation for Connecting to WI-FI Network

No. of Users	Average (in sec)	St Deviation (in sec)	Median (in sec)
1	3.45	-	3.45
2	3.8	0.438	3.8
3	3.976	0.271	3.82
4	4.065	0.172	4.02
5	4.392	1.325	4.23
6	4.531	1.632	4.325

By referring to the table 6.2, we can observe that there is a huge increment in Standard Deviation for the users 4. Immediately, while connecting the network to 5 users, there is a short decrement in Standard Deviation. Immediately, the Standard Deviation goes on increasing, when 6 users are connecting to the same network.

Figure 6.2 Average Time Taken to Open the Application



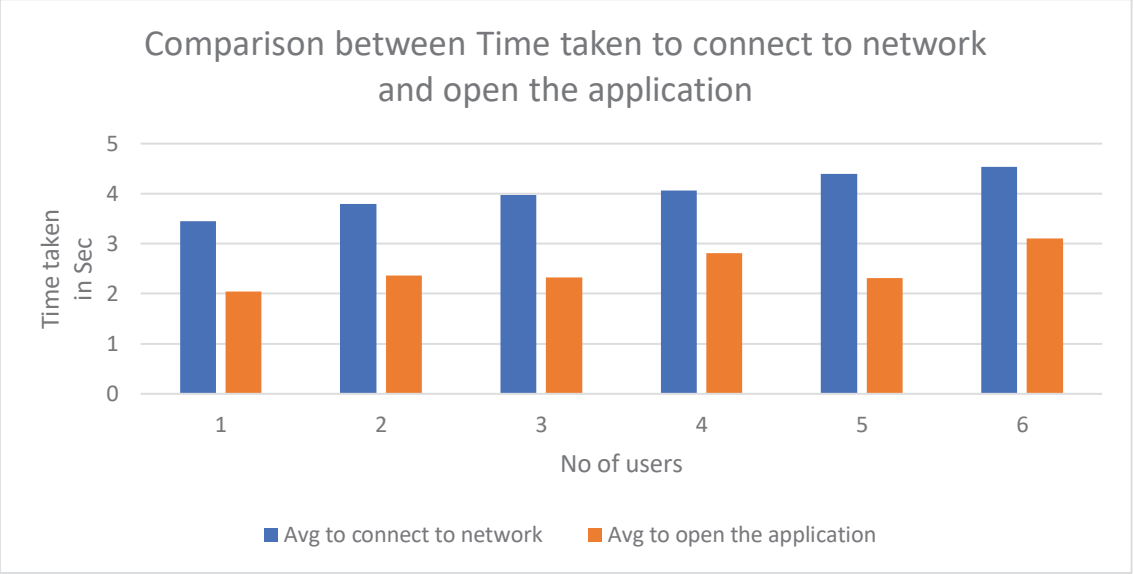
By referring to the graph 6.2, we can observe that by increasing the number of users, the time taken to open the application almost remains constant. For the last couple of values, there is some variability in graph. But while doing experiment for 4 users, the time taken to open the application is high.

Table 6.3 Average, Median and Standard Deviation to Open the Application

No. of Users	Average (in sec)	St Deviation (in sec)	Median (in sec)
1	2.05	-	2.05
2	2.36	0.353	2.36
3	2.33	0.408	2.19
4	2.812	1.195	2.35
5	2.31	0.268	2.19
6	3.111	0.624	3.055

By referring to the table 6.3, we can observe that there is a huge increment in Standard Deviation for the users 4. Immediately, while using the application for 5 users, there is a huge decrement in Standard Deviation. Immediately, the Standard Deviation goes on increasing, when 6 users are using the application at the same time.

Figure 6.3 Comparison between Time taken to connect to network and open the application



From the above graph 6.3, we can observe that for any number of users the time taken to connect to WI-FI network is high compared to the time taken to open and use the application. It means there is an effect in the network side. The performance of any application also depends upon the network side.

6.2 AUTOMATED TESTING OF EDMODO APPLICATION

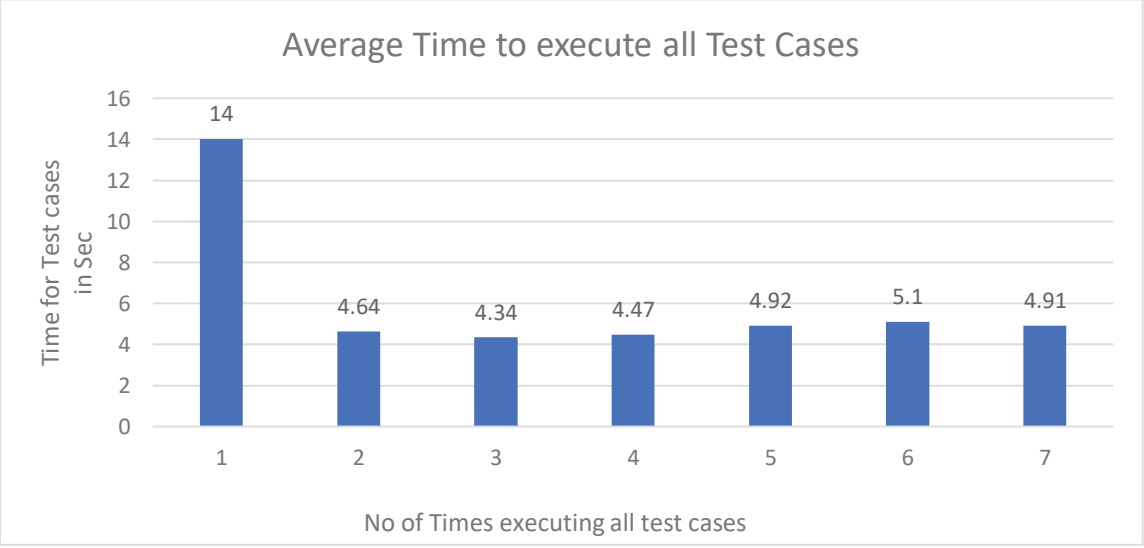
Table 6.4 Time Difference between test cases (in seconds)

No. of Times	Test Case T2-T1	Test Case T3-T2	Test Case T4-T3	Test Case T5-T4	Test Case T6-T5	Test Case T7-T6
1	26	12	1	3	49	6
2	9	5	0.5	1	12	3
3	9	4	0.4	1	11	2
4	9	4	0.3	1	11	2
5	9	5	1	0.5	12	2
6	9	4	0.5	0.2	15	1
7	8	5	0.4	1	12	1

Where T1=0

T1 = Connect to Browser, T2 = Connect to URL, T3 = Click on login button, T4 = Enter the login credentials, T5 = Enter the message, T6 = Enter the Assignment in group, T7 = Click on logout button.

Figure 6.4 Difference between the test cases



By referring to the table and graph 6.4, the average difference between the time taken from succeeding test case to previous test case has been calculated. We can observe that the time taken to run each test case for the first time is high compared to remaining. From 2nd to 7th time, it completes less time to execute all test cases. But for the first time, it needs more time to complete the task. The specific reason for performing automated testing up to 7 times is because of constant time taken to execute all those test cases. From this, we can understand that the performance of any application decreases in the initial stage (running the application for the first time).

6.3 PERFORMANCE TESTING OF IPAD APPLICATION

In performance testing, all the measurements have been performed through Results Tree table. View results tree is a combination of all measurements displayed by each sample. While testing, it displays the complete overview of every requested sample. It also displays listeners and assertions. For Results Tree in all experiments, sample count = 1 millisecond and Error count = 0. For some validations, we had included 4 background applications. The following applications used in our experiment are Google Chrome, Internet Explorer on the windows machine and Safari, Google Chrome on iPad.

For the performance testing, we had calculated load time, latency, connect time, size in bytes, sent bytes, header size and body size. Here

Load Time is the time difference between when the request was sent and when the response has completely received.

Latency is the time difference between when the request was sent and time when response has started to receive.

Connect time is the time taken to establish the connection.

Header size in bytes is the size of page headers in bytes.

Body size in bytes is the size of page body in bytes.

Size in bytes is the total page size in bytes.

Size in bytes = Header size in bytes + body size in bytes.

The readings mentioned in all the below tables are taken for 1 sample. As we are using the Apache JMeter tool, it can manage maximum of 10000 users. While testing the performance of the application, we had included load test (up to 10000 users) in parallel.

View results tree is a listener for better understanding the results of each sample.

For each validation, the experiment is executed for 10 times. As the performance of the application decreases for every 5-6 frequent executions (mentioned by company). So, the experiment is executed for 10 times. The below readings are the average values.

6.3.1 VALIDATION 1

Test Case with network A, No background applications & Login Average of View Result Tree

Table 6.5 Average of View Result Tree of Login function for Validation 1

No. of Users	Load Time (ms)	connect Time (ms)	Latency (ms)	size in bytes	sent bytes	Header size	Body size
1	229	22	229	4354	293	604	3750
2	1927	69	1117	632846	401	416	632430
3	1780	70	1014	632846	401	416	632430
4	1643	89	912	632846	401	416	632430
5	1714	69	946	632846	401	416	632430
10	2197	72	1425	632846	401	416	632430
20	2028	74	1218	632846	401	416	632430
30	1898	73	1115	632846	401	416	632430
100	1730	70	910	632846	401	416	632430
1000	1794	95	1013	632846	401	416	632430
10K	1776	73	952	632846	401	416	632430

The above table 6.5 is a View Results Tree Listener which contains the data of first succeeded sampler. It displays the complete details of http request and response. The above results are completely in the text format. The testing has been performed for maximum of 10000 users. By referring to the table 6.5, we can observe that the sampler with 1 user is completely irrelevant to the remaining samples. It shows that there is less data for size in bytes, sent bytes, body size, connect time, load time, latency and increase in header size. This is because of low response data. When the JMeter hits the targeted server for the first time, the response hasn't received completely. So, it displays with low connect time and less size in bytes, body size.

We can also observe that the size in bytes, sent bytes, header size and body size are similar for the remaining samples. As the first request has already been sent, the server gets activated and responds with the actual size of header, body, sent bytes and size in bytes.

Test Case with network A, no background applications & Run the application Average of View Result Tree

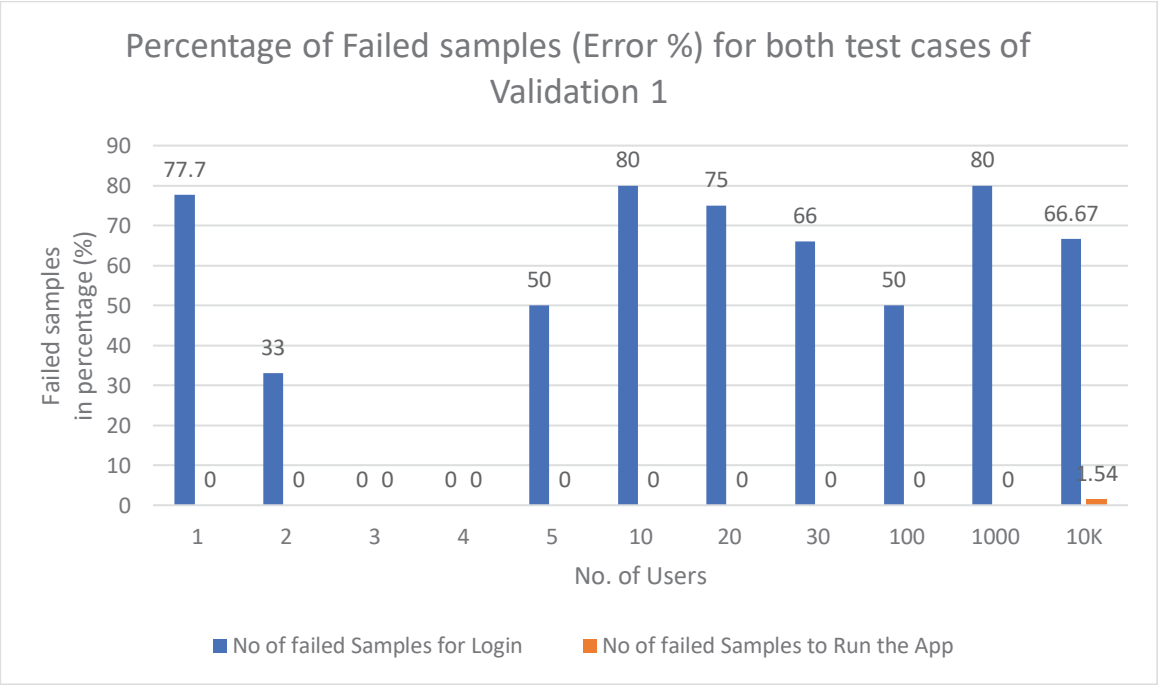
Table 6.6 Average of View Result Tree of Run Application for Validation 1

No. of Users	Load Time (ms)	connect Time (ms)	Latency (ms)	size in bytes	sent bytes	Header size	Body size
1	295	46	295	909	509	416	493
2	313	47	313	441	611	416	25
3	327	73	327	865	648	416	449
4	350	74	350	865	645	416	449
5	328	69	328	865	647	416	449
10	298	77	298	865	648	416	449
20	236	71	236	865	599	416	449
30	342	76	342	865	648	416	449
100	363	89	363	865	648	416	449
1000	336	73	336	865	647	416	449
10K	350	72	350	865	647	416	449

The above table 6.6 is a View Results Tree Listener which contains the data of first succeeded sampler. It displays the complete details of http request and response. The above results are completely in the text format. The testing has been performed for maximum of 10000 users. By referring to the table 6.6, we can observe that the body size of the sample with 2 users is too low compared to the remaining samples. It is because of the error (failed requests) within overall requested samples of 2 users. There is no overall success ratio of the requested samples with 2 users.

As the test case is to run the application, we can observe low size in bytes and body size. It is because of using the application. As the tester is playing the game, he should shift to different pages in the application. As change in the pages, the body size and size in bytes are low compared to the previous test case (login).

Figure 6.5 Comparing both the test cases of Validation 1



By referring to the above graph 6.5, comparison of both the test cases present in the Validation 1 has been taken. The error percentage is in the Summary Report located in Appendix Section. We can understand that the error percentage is high for the login function compared to running the application. It is because of increase in the number of failure requests for the login test case compared to the other test case. When the JMeter hits the targeted server for the first time, the response time received for the login function is exceeded, resulting in a time-out.

6.3.2 VALIDATION 2

Test Case with network B, No background applications & Login Average of View Result Tree

Table 6.7 Average of View Result Tree of Login function for Validation 2

Users	Load Time (ms)	connect Time (ms)	Latency (ms)	size in bytes	sent bytes	Header size	Body size
1	2139	83	1128	441	401	416	632474
2	2131	84	1122	632890	401	416	632474
3	2050	85	1012	632890	401	416	632474
4	2127	87	1208	632890	401	416	632474
5	2075	84	1190	7724	401	763	632474
10	2226	64	1600	632890	401	416	632474
20	1968	98	997	632890	401	416	632474
30	3081	95	1004	632890	401	416	632474
100	2261	97	1221	632890	401	416	632474
1000	2031	87	968	632891	401	416	632474
10K	2198	85	1120	632891	401	416	632475

The above table 6.7 is a View Results Tree Listener which contains the data of first succeeded sampler. It displays the complete details of http request and response. The above results are completely in the text format. The testing has been performed for maximum of 10000 users. By referring to the table 6.7, we can observe that there is a less size in bytes for 1 user and 5 users. It is because of the change in the network and the server haven't responded to the corresponding request. So, it responds with low size in bytes.

Test Case with network B, no background applications & Run the application Average of View Result Tree

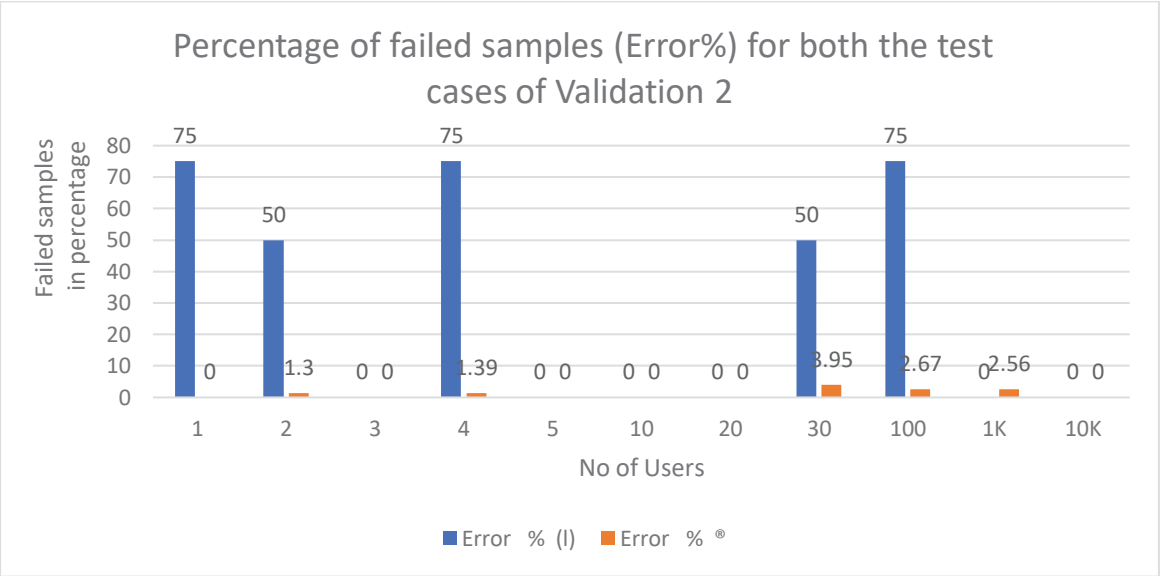
Table 6.8 Average of View Result Tree of Run Application for Validation 2

No. of Users	Load Time (ms)	connect Time (ms)	Latency (ms)	size in bytes	sent bytes	Header size	Body size
1	409	68	409	865	596	416	449
2	440	79	440	865	647	416	449
3	430	80	430	865	647	416	449
4	447	51	447	865	596	416	449
5	444	83	444	865	647	416	449
10	492	88	492	865	647	416	449
20	467	67	467	865	647	416	449
30	412	85	412	865	647	416	449
100	485	98	485	865	647	416	449
1000	427	94	427	865	647	416	449
10K	444	96	444	865	647	416	449

The above table 6.8 is a View Results Tree Listener which contains the data of first succeeded sampler. It displays the complete details of http request and response. The above results are completely in the text format. The testing has been performed for maximum of 10000 users. By referring to the table 6.8, we can observe that there is no much increment or decrement in the samples for all the users. The test case is to run the application. So, there will be change in the pages, the body size and size in bytes are low compared to the previous test case (login).

Comparing both the test cases of Validation 2

Figure 6.6 Comparing both the test cases of Validation 2



By referring to the above graph 6.6, comparison of both the test cases present in the Validation 2 has been taken. The error percentage is in the Summary Report located in Appendix Section. We can understand that the error percentage is high for the login function compared to running the application. It is because of increase in the number of failure requests for the login test case compared to the other test case. Especially for user 1,2,4,30,100 of login function, when the JMeter hits the targeted server, the response time received for the login function is exceeded, resulting in a time-out.

6.3.3 VALIDATION 3

Test Case with network A, 4 background applications & Login Average of View Result Tree

Table 6.9 Average of View Result Tree of Login function for Validation 3

Users	Load Time (ms)	connect Time (ms)	Latency (ms)	size in bytes	sent bytes	Header size	Body size
1	2221	91	1449	632846	401	416	632846
2	1790	73	1014	632846	401	416	632846
3	1824	75	997	632846	401	416	632846
4	1887	68	1118	632846	401	416	632846
5	1740	73	962	632846	401	416	632846
10	1797	93	1014	632846	401	416	632846
20	1943	73	1123	632846	401	416	632846
30	2051	73	1191	632846	401	416	632846
100	1762	71	1034	632846	401	416	632846
1000	2038	126	1220	632846	401	416	632846
10K	1970	72	1246	632846	401	416	632846

The above table 6.9 is a View Results Tree Listener which contains the data of first succeeded sampler. It displays the complete details of http request and response. The above results are completely in the text format. The testing has been performed for maximum of 10000 users. By referring to the table 6.9, we can observe that there is no much increment or decrement in the results. Although there is a heavy load on the application the results are same. For this test case, it shows that there is no external workload affected to this application.

Test Case with network A, 4 background applications & Run the application Average of View Result Tree

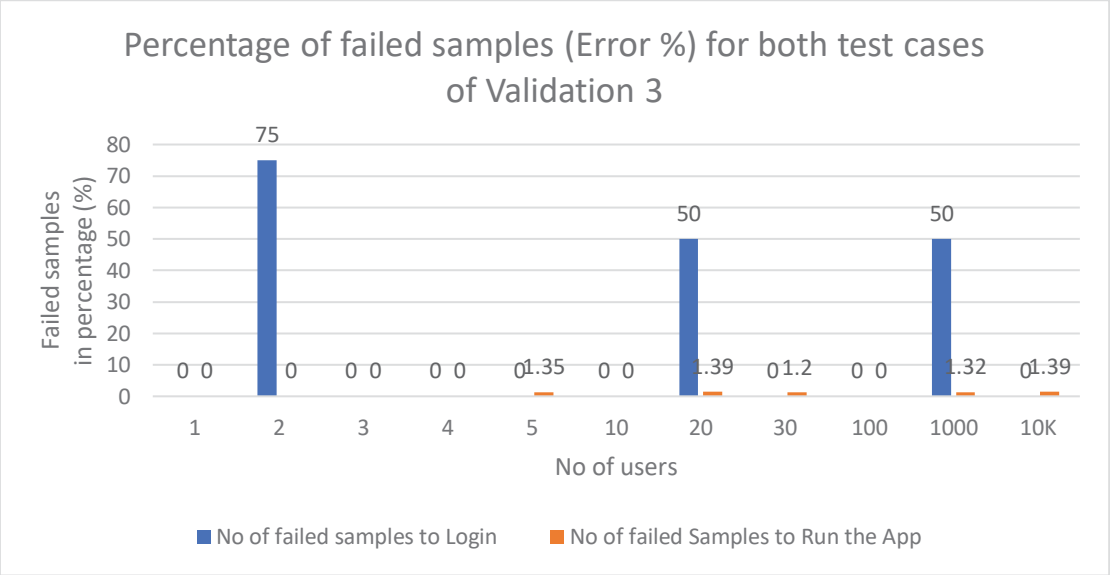
Table 6.10 Average of View Result Tree of Run Application for Validation 3

Users	Load Time (ms)	connect Time (ms)	Latency (ms)	size in bytes	sent bytes	Header size	Body size
1	384	72	384	865	647	416	449
2	320	75	320	865	647	416	449
3	358	74	358	865	647	416	449
4	320	42	320	441	647	416	25
5	332	52	332	865	647	416	449
10	362	81	362	865	647	416	449
20	369	72	369	865	647	416	449
30	362	72	362	865	647	416	449
100	259	71	259	865	647	416	449
1000	390	72	390	865	647	416	449
10K	362	70	362	865	647	416	449

The above table 6.10 is a View Results Tree Listener which contains the data of first succeeded sampler. It displays the complete details of http request and response. The above results are completely in the text format. The testing has been performed for maximum of 10000 users. By referring to the table 6.10, we can observe that the body size of the sample with 4 users is too low compared to the remaining samples. It is because of the error within overall requested samples of 4 users. There is no overall success ratio of the requested samples with 4 users. Although there is a heavy load on the application the results are same. For this test case, it shows that there is no external workload affected to this application.

Comparing both the test cases of Validation 3

Figure 6.7 Comparing both the test cases of Validation 3



By referring to the above graph 6.7, comparison of both the test cases present in the Validation 3 has been taken. The error percentage is in the Summary Report located in Appendix Section. We can understand that the error percentage is high for the login function compared to running the application. It is because of increase in the number of failure requests for the login test case compared to the other test case. Especially for user 2,20,1000 of login function, when the JMeter hits the targeted server, the response time received for the login function is exceeded, resulting in a time-out.

6.3.4 VALIDATION 4

Test Case with network B, 4 background applications & Login Average of View Result Tree

Table 6.11 Average of View Result Tree of Login function for Validation 4

Users	Load Time (ms)	connect Time (ms)	Latency (ms)	size in bytes	sent bytes	Header size	Body size
1	2221	91	1449	632846	401	416	632846
2	1790	73	1014	632846	401	416	632846
3	1824	75	997	632846	401	416	632846
4	1887	68	1118	632846	401	416	632846
5	1740	73	962	632846	401	416	632846
10	1797	93	1014	632846	401	416	632846
20	1943	73	1123	632846	401	416	632846
30	2051	73	1191	632846	401	416	632846
100	1762	71	1034	632846	401	416	632846
1000	2038	126	1220	632846	401	416	632846
10K	1970	72	1246	632846	401	416	632846

The above table 6.11 is a View Results Tree Listener which contains the data of first succeeded sampler. It displays the complete details of http request and response. The above results are completely in the text format. The testing has been performed for maximum of 10000 users. By referring to the table 6.11, we can observe that there is no much increment or decrement in the results. Although there is a heavy load on the application the results are same. For this test case, it shows that there is no external workload affected to this application.

Test Case with network B, 4 background applications & Run the application Average of View Result Tree

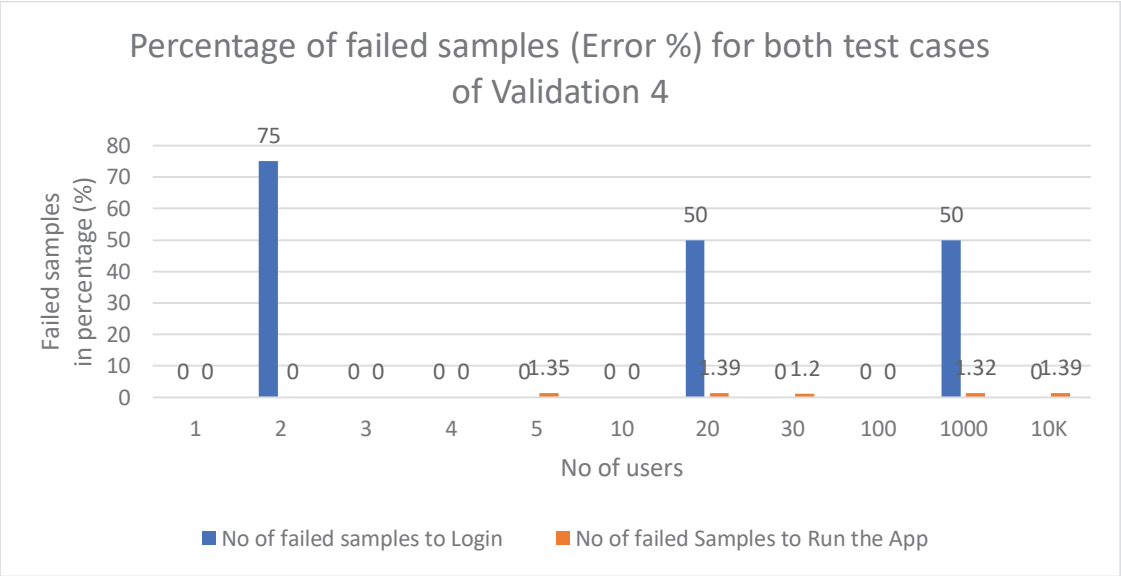
Table 6.12 Average of View Result Tree of Run Application for Validation 4

Users	Load Time (ms)	connect Time (ms)	Latency (ms)	size in bytes	sent bytes	Header size	Body size
1	403	63	403	865	660	416	449
2	363	81	363	865	647	416	449
3	423	80	423	865	647	416	449
4	404	91	404	865	596	416	449
5	306	85	306	865	647	416	449
10	391	65	391	865	647	416	449
20	257	79	257	865	647	416	449
30	390	86	390	865	647	416	449
100	535	85	535	441	4500	416	25
1000	445	82	445	865	647	416	449
10K	444	100	444	865	647	416	449

The above table 6.12 is a View Results Tree Listener which contains the data of first succeeded sampler. It displays the complete details of HTTP request and response. The above results are completely in the text format. The testing has been performed for maximum of 10000 users. By referring to the table 6.12, we can observe that the body size of the sample with 100 users is too low compared to the remaining samples. It is because of the error within overall requested samples of 100 users. There is no overall success ratio of the requested samples with 100 users. We can also observe that the size in bytes for 100 users are more compared to the remaining samples. It is because of sending more requests and receiving most of the requests at the same time. It also affects the low body size. Although there is a heavy load on the application the results are same. For this test case, it shows that there is no external workload affected to this application.

Comparing both the test cases of Validation 4

Figure 6.8 Comparing both the test cases of Validation 4



By referring to the above graph 6.8, comparison of both the test cases present in the Validation 4 has been taken. The error percentage is in the Summary Report located in Appendix Section. We can understand that the error percentage is high for the login function compared to running the application. It is because of increase in the number of failure requests for the login test case compared to the other test case. Especially for user 2,20,1000 of login function, when the JMeter hits the targeted server, the response time received for the login function is exceeded, resulting in a time-out.

6.3.5 COMPARISON OF ERROR PERCENTAGE FOR ALL VALIDATIONS

Table 6.13 Comparison of failed samples (error %) for all validations

No. of Validations	Validations	Test cases	Avg of failed samples (ERROR percentage)
1	Validation 1	Network A, no background applications, Login to the iPad application	52.57
2	Validation 1	Network A, no background applications, Run the iPad application	0.14
3	Validation 2	Network B, no background applications, Login to the iPad application	29.54
4	Validation 2	Network B, no background applications, Run the iPad application	1.07
5	Validation 3	Network A, 4 background applications, Login to the iPad application	15.90
6	Validation 3	Network A, 4 background applications, Run to the iPad application	3.9
7	Validation 4	Network B, 4 background applications, Login to the iPad application	15.90
8	Validation 4	Network B, 4 background applications, Run the iPad application	0.60

By referring to the table 6.13, we can observe the average of error percentage for all the validations. It shows that there is a huge difference between the error percentage of the following test cases (login and run the application).

For login test case, the requested samples are sent until the application gets logged in. For suppose, if we get the response after 2 requested samples then the testing is stopped because there is no much work to do while logging in. The requested samples are noted in the Summary Report of Appendix Section.

For run the application test case, the application is tested for maximum of 4 minutes. So, the requested samples are more compared to the login test case. The requested samples are noted in the Summary Report of Appendix Section.

From all the above validations and results, we can conclude that the error percentage is high for login test case. Even though there is a change in the network situations and workload, the error percentage remains same. It means the percentage of failed requests are high for login test case. The performance of the application decreases while logging into the application. Once the user gets logged in, the application works well.

Chapter 7

RESEARCH QUESTIONS & ANSWERS

7.1 RESEARCH QUESTIONS & ANSWERS

1. What are the commonly perceived problems seen from performing Manual Testing of mobile application?

The graphs in the result section for Manual Testing shows detailed information about the time taken for both the test cases. If we observe carefully from the results table and graphs of both the test cases, we can clarify that, by increasing the users, the time taken to connect to a WI-FI network is high compared to the time taken for opening the application. By referring to the table 6.2, we can observe that there is a huge increment in Standard Deviation for the users 4. Immediately, while connecting the network to 5 users, there is a short decrement in Standard Deviation. Immediately, the Standard Deviation goes on increasing, when 6 users are connecting to the same network. But from the table 6.3, we can observe that there is no much increment or decrement in all values for increasing the users. By this we can conclude that the performance of any application has also an impact on the network side. As increasing the users, the time taken to connect to network is high compared to opening the application.

2. What are the commonly perceived problems seen from performing Automated Testing of an Edmodo Application?

The graphs in the result section for Automated testing shows the most common perceived problems seen from performing Automated Testing. If we observe carefully from the results table and graphs of all the test cases, the time taken to execute all the test cases for the first time is high compared to the time taken to execute all the test cases for remaining times. By referring to the table 6.4, testing in between 2nd to 7th time, it needs less time to complete all tasks. But for the first time, it needs more time to complete the task. It means whenever we open an application for the first time, the delay time is high compared to the remaining times.

3. What are the strategies that are required to prevent an iPad application from slowing down when a large number of users were using the application at the same time?

The most important strategy to prevent an iPad application is to periodically test the performance of the application. By dividing the test cases, based on the working of the application, we can find the actual problem and can prevent the application from slowing down. Because of the test cases, it is possible to find the actual error located in the

application. The results in section 6.3 clearly explain about the strategies required to prevent the application. The application's performance also depends upon device performance and network performance. So, it is better to recheck the device and network conditions. By referring to the tables and graphs present in the section 6.3, we can understand that the error percentage is high for the login function compared to running the application. It is because of increase in the number of failure requests for the login test case compared to the other test case. As we sent a lot of samples for the first test case, the received samples are very low. It means the percentage of failed http requests are high for login test case.

Chapter 8

CONCLUSION & FUTURE WORK

8.1 CONCLUSION

The summary of the thesis work is provided in this section. The main aim of the thesis is to test the performance of an iPad application.

A mobile application is tested manually (usability testing), by increasing the number of users under only one wireless network. As per section 5.1, the comparison has been done for both the test cases (connect to WI-FI and run the app). As increasing the number of users, the time taken for connecting to WI-FI is low compared to opening and using the application.

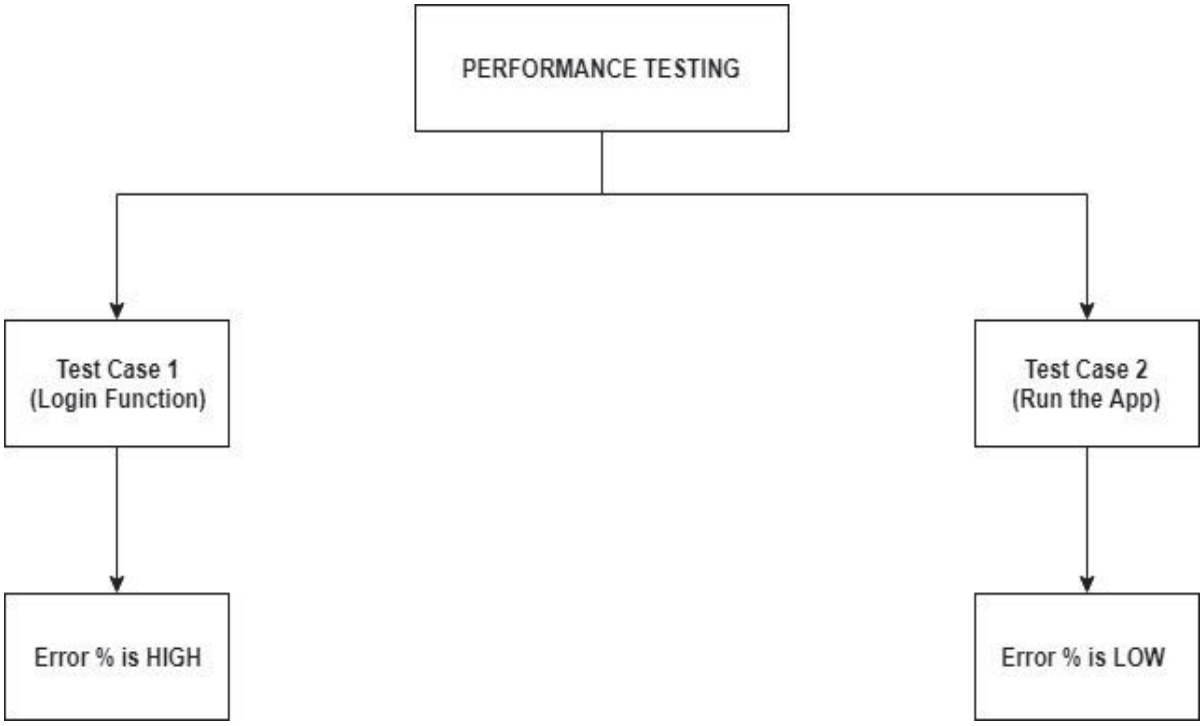
This research also provides the automated testing of a web application. By understanding the application, some test cases have been initialized. A script has been written to process the test automatically. The difference between the time taken from the succeeding test case to the previous test case has been calculated. The comparisons have been taken for each time when we run the script for automated testing. It is found that the time taken to run each test case for the first time is high compared to remaining.

The performance metrics of an iPad application has been calculated. As per the section 4.1.3, we had divided it into two test cases by understanding the application. The testing has been done under various situation i.e. different network conditions, heavy load on users and applications running in the background. An important strategy to prevent the application from slowing down is to periodically test the performance of the application. It is also better to recheck the device and network performance. Based on the above results and validations of performance testing, we can conclude that the error percentage is very high for the first test case (i.e. login function) compared to the second test case (run the app). As we sent a lot of samples for the first test case, the received samples are very low. It means the percentage of failed requests are high for login test case. As testing is taken for more than 200 times, the result is the same.

But once the application is started running, the application works perfectly well. The error percentage is very low. Even though if the load is high (10,000 users), the application works very fast.

Summarizing the observation for performance testing has been shown clearly in the below diagram.

Figure 8.1 Conclusion for Performance Testing



8.2 FUTURE WORK

If an application is developed, most of them think about the quality but performance makes a huge impact. If there is an issue with performance, it is better to use DevOps [27] strategy. It means the performance of any application is tested while developing the code functionality itself. Whenever there is any fix or release of the application, the time to the users should be minimum. So continuous delivery and continuous deployment [28] helps a lot to increase the performance.

In some cases, if the device gets updated then there is a possibility of decrease in the performance of the application. So, the application must be periodically rechecked. As automation is the future, there will be new testing tools and methods. So, when the application is in a developing stage or developed stage, it is better to check the performance of the application using those tools.

Chapter 9

REFERENCES

[1] S. Ickin, *Quality of experience on smartphone s : network, application, and energy perspectives*.

Karlskrona: Department of Communication Systems, Blekinge Institute of Technology, 2015.

[2] S. Afshari and N. Movahhedinia, "QoE assessment of interactive applications in computer networks," *M ultimed. Tools Appl.*, vol. 75, no. 2, pp. 903–918, Jan. 2016.

[3] C. Lorentzen, M. Fiedler, H. Johnson, J. Shaikh, and I. Jørstad, "On user perception of web login — A study on QoE in the context of security," in *2010 Australasian Telecommunication Networks and Applications Conference*, 2010, pp. 84–89.

[4] L. Panizo, A. Díaz, and B. García, "Model-based testing of apps in real network scenarios," *Int. J . Softw. Tools Technol. Transf.*, Apr. 2019.

[5] F. Dias and A. C. R. Paiva, "Pattern-Based Usability Testing," in *2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2017, pp. 366–371.

[6] A. Antonova and B. Shanovskiy, "RESEARCH AND ANALYSIS OF APPLICATION OF AUTOMATED TESTING IN WEB APPLICATIONS," *Avtom. Tehnol. Bizn.-Process .*, vol. 10, no. 1, Apr. 2018.

[7] "Exploiting load testing and profiling for Performance Antipattern Detection - ScienceDirect," 19-May-2019. [Online]. Available: https://www-sciencedirect-com.miman.bib.bth.se/science/article/pii/S0950584917302276. [Accessed: 19-May-2019].

[8] S. Pradeep and Y. K. Sharma, "A Pragmatic Evaluation of Stress and Performance Testing Technologies for Web Based Applications," in *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 399–403.

[9] "Learning Selenium Testing Tools [Video]," 19-May-2019. [Online]. Available: https://www.oreilly.com/library/view/learning-selenium-testing/9781783985562/?ar. [Accessed: 19-May-2019].

[10] C. Hursen and F. G. Fasli, "The Impact of Reflective Teaching Applications Supported by Edmodo on Prospective Teachers' Self-Directed Learning Skills," *Int. J. Emerg. Technol. Learn. IJE T*, vol. 12, no. 10, pp. 21–34, Nov. 2017.

[11] "Performance Testing with JMeter 3 - Third Edition av Bayo Erinle (Häftad)," *Bokus.co m*, 19-May-2019. [Online]. Available: https://www.bokus.com/bok/9781787285774/performance-testing-with-jmeter-3-third-edition/. [Accessed: 19-May-2019].

[12] S. Duttagupta, R. Virk, and M. Nambiar, "Software bottleneck analysis during performance testing," in *2015 International Conference and Workshop on Computing and Communication (IEMCON)*, 2015, pp. 1–7.

[13] "Pro iOS Apps Performance Optimization [Book]," 19-May-2019. [Online]. Available: https://www.oreilly.com/library/view/pro-ios-apps/9781430237174/?ar. [Accessed: 19-May-2019].

[14] "Pro Apache JMeter: Web Application Performance Testing [Book]," 19-May-2019. [Online]. Available: https://www.oreilly.com/library/view/pro-apache-jmeter/9781484229613/?ar. [Accessed: 19-May-2019].

[15] W. Barth, *N agios: System and Network Monitoring*. 2008.

[16] A. Balachandran *et al .*, "Modeling Web Quality-of-experience on Cellular Networks," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, New York, NY, USA, 2014, pp. 213–224.

[17] A. M. F. V. de Castro, G. A. Macedo, E. F. Collins, and A. C. Dias-Neto, “Extension of Selenium RC tool to perform automated testing with databases in web applications,” in *2013 8 th International Workshop on Automation of Software Test (AST)*, 2013, pp. 125–131.

[18] “Mastering Selenium WebDriver [Book],” 19-May-2019. [Online]. Available: https://www.oreilly.com/library/view/mastering-selenium-webdriver/9781784394356/?ar. [Accessed: 20-May-2019].

[19] “Mastering Selenium WebDriver [Book],” 19-May-2019. [Online]. Available: https://www.oreilly.com/library/view/mastering-selenium-webdriver/9781784394356/. [Accessed: 20-May-2019].

[20] E. H. Halili, *A pache JMeter: A Practical Beginner’s Guide to Automated Testing and Performance Measurement for Your Websites*. Packt Publishing Ltd, 2008.

[21] G. A. Cayton-Hodges, G. Feng, and X. Pan, “Tablet-Based Math Assessment: What Can We Learn from Math Apps?,” *Educ. Technol. Soc .*, vol. 18, no. 2, pp. 3–20, 2015.

[22] “Nagios 3 Enterprise Network Monitoring: Including Plug-Ins and Hardware Devices - Max Schubert, Derrick Bennett, Jonathan Gines, Andrew Hay, John Strand - Häftad (9781597492676) | Bokus,” 19-May-2019. [Online]. Available: https://www.bokus.com/bok/9781597492676/nagios-3-enterprise-network-monitoring-including-plug-ins-and-hardware-devices/. [Accessed: 20-May-2019].

[23] “Hands-On Automation Testing with Java for Beginners | PACKT Books,” 20-May-2019. [Online]. Available: https://www.packtpub.com/application-development/hands-automation-testing-java-beginners. [Accessed: 20-May-2019].

[24] M. I. Malkawi, “The art of software systems development: Reliability, Availability, Maintainability, Performance (RAMP),” *H um.-Centric Comput. Inf. Sci.*, vol. 3, no. 1, p. 22, Dec. 2013.

[25] M. L. Hutcheson, “Software testing fundamentals - methods and metrics,” 2003.

[26] S. Jansen and A. McGregor, “Performance, Validation and Testing with the Network Simulation Cradle,” in *a nd Simulation 14th IEEE International Symposium on Modeling, Analysis*, 2006, pp. 355–362.

[27] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, “DevOps,” *IEEE Softw .*, vol. 33, no. 3, pp. 94–100, May 2016.

[28] K. Quibeldey-Cirkel and C. Thelen, “Continuous Deployment,” *I nform.-Spektrum*, vol. 35, no. 4, pp. 301–305, Aug. 2012.

[29] “What is Apache Web Server? - Definition from Techopedia,” 20-May-2019. [Online]. Available: https://www.techopedia.com/definition/4851/apache-web-server. [Accessed: 20-May-2019].

[30] “Instant Nagios Starter [Book],” 20-May-2019. [Online]. Available: https://www.oreilly.com/library/view/instant-nagios-starter/9781782162506/. [Accessed: 20-May-2019].

Chapter 10

APPENDIX

In this chapter, it shows the Averages of Summary Report and Aggregate Report. It provides more results in detailed version for all the above validations.

Summary Report is one of the listeners presents in JMeter. It contains group of values for every request in the test. The main advantage of Summary Report is it consumes less memory compared to the remaining listeners.

Average is the average time taken to execute every http request.

Minimum is the shortest time taken by sample for the request.

Maximum is the longest time taken by sample for the request.

Error % is the percentage of failed requests.

Throughput is the number of requests that are processed per unit time by the server. it is measured in kb/sec.

In Summary Report for login test case, the requested samples are sent until the application gets logged in. But for running the application, the requested samples are sent until the completion of that test.

Aggregate Report is one of the listeners presents in JMeter. It contains few more parameters which are not present in Summary Report and View Results Tree. They are Median, 90%-line, 95% line and 99% line. The results provided in this section are the averages of Average Report.

Median is the time in the middle of a set of samples result.

90% line means 90% of samples took no more than this time (mentioned time).

95% line means 95% of samples took no more than this time (mentioned time). 99%

line means 99% of samples took no more than this time (mentioned time).

Appendix A

Test Case with network A, no background applications & Login

Average of Summary Report

A 1 Summary Report of Login function for validation 1

users	samples	Avg	Min	Max	St Dev	Error %	Throughput (sec)	Received (kb/sec)	sent (kb/sec)	Avg bytes	CI
1	18	154	0	1705	400.97	77.7	6	211.19	3.37	36030.08	199.39
2	3	1224	0	1927	869.08	33	4.1	28.48	0.02	422146.7	2158.97
3	1	1780	1780	1780	-	0	33.7	347.20	0.22	632846.0	-
4	1	1643	1643	1643	-	0	36.5	376.15	0.24	632846.0	-
5	2	857	0	1714	857	50	1.2	360.99	0.23	316797	7699.83
10	5	439	0	2197	878.80	80	2.3	282.63	0.18	127167.6	1091.17
20	4	507	0	2028	878.15	75	2.6	305.82	0.19	158772.5	1398.36
30	3	632	0	1898	894.73	66	1.6	326.38	0.21	211447.3	2222,62
100	2	865	0	1730	865	50	1.2	357.66	0.23	316797.0	7771.71
1K	5	358	0	1794	717.6	80	2.8	346.12	0.22	127167.6	891.01
10K	3	592	0	1776	837.21	66.67	1.7	348.80	0.22	211447.3	2079.74

Average of Aggregate Report

A 2 Aggregate report of Login function for Validation 1

Users	Median	90% line	95% line	99% line
1	0	371	476	1705
2	1747	1927	1927	1927
3	1780	1780	1780	1780
4	1643	1643	1643	1643
5	0	1714	1714	1714
10	0	0	2197	2197
20	0	2028	2028	2028
30	0	1893	1893	1893
100	0	1730	1730	1730
1K	0	0	1794	1794
10K	0	1776	1776	1776

Test Case with network A, no background applications & Run the application

Average of Summary Report

A 3 Summary Report of Run the app for validation 1

users	samples	Avg	Min	Max	St Dev	Error %	Throughput (sec)	Received (kb/sec)	sent (kb/sec)	Avg bytes	CI
1	38	342	177	2967	434.65	0	43.2	12.14	0.49	17280.6	142.86
2	34	339	174	1661	234.23	0	43.2	13.46	0.56	19132.6	81.72
3	30	349	204	1893	294.42	0	41.6	14.64	0.56	21607.9	109.93
4	75	293	172	1755	178.96	0	45.2	6.62	0.52	8990.8	41.174
5	79	276	164	648	70.02	0	1.4	0.79	0.92	574.3	15.68
10	26	329	169	1833	304.32	0	37.5	15.16	0.52	24845.4	122.91
20	48	300	171	610	7151	0	39.6	0.37	0.49	577.5	2076.43
30	26	347	183	1724	281.08	0	32.8	13.26	0.46	24845.9	113.53
100	70	283	169	827	91.29	0	1.1	0.62	0.71	583.5	21.76
1K	73	314	178	1971	204.68	0	1.4	12.49	0.97	9219.8	47.75
10K	65	282	0	413	68.42	1.54	1.9	1.10	1.27	580.2	16.95

Average of Aggregate Report

A 4 Aggregate Report of Run the App for validation 1

Users	Median	90% line	95% line	99% line
1	281	333	355	2967
2	308	349	350	1661
3	289	397	400	1893
4	285	343	350	452
5	280	328	350	439
10	298	323	343	1833
20	305	334	409	610
30	298	342	431	1724
100	298	348	352	456
1K	298	363	399	445
10K	293	350	380	901

Appendix B

Test Case with network B, no background applications & Login

Average of Summary Report

B 1 Summary Report of Login function for validation 2

users	Samples	Avg	Min	Max	St Dev	Error %	Throughput (sec)	Received (kb/sec)	sent (kb/sec)	Avg bytes	CI
1	4	534	0	2139	926.21	75	1.9	289.97	0.18	158783.5	1473.80
2	2	1065	0	2131	1065.5	50	56.3	290.37	0.18	316819	9573.13
3	1	2050	2050	2050	0	0	29.3	301.49	0.19	632891	-
4	4	531	0	2127	921.02	75	1.9	291.61	0.18	158783	1465.54
5	1	2075	2075	2075	0	0	28.9	297.86	0.19	632891	-
10	1	2226	22260	2226	0	0	27	277.65	0.18	632891	-
20	1	1968	1968	1968	0	0	30.5	314.05	0.20	632891	-
30	2	1540	0	3081	1540.5	50	38.9	200.84	0.13	316819	13840.84
100	4	565	0	2261	979.04	75	1.8	274.32	0.17	158783.5	1557.87
1K	1	2031	2031	2031	0	0	29.5	304.31	0.19	632891	-
10K	1	2198	2198	2198	0	0	27.3	281.19	0.18	632891	-

Average of Aggregate Report

B 2 Aggregate Report of Login function for validation 2

Users	Median	90% line	95% line	99% line
1	0	2139	2139	2139
2	0	2131	2131	2131
3	2050	2050	2050	2050
4	0	2127	2127	2127
5	2075	2075	2075	2075
10	2226	2226	2226	2226
20	1968	1968	1968	1968
30	0	3081	3081	3081
100	0	2261	2261	2261
1K	2031	2031	2031	2031
10K	2198	2198	2198	2198

Test Case with network B, no background applications & Run the application
Average of Summary Report

B 3 Aggregate Report of Run the App for validation 2

users	Samples	Avg	Min	Max	St Dev	Error %	Throughput (sec)	Received (kb/sec)	sent (kb/sec)	Avg bytes	CI
1	74	381	221	620	72.71	0	1.01	0.57	0.70	582.3	16.84
2	75	367	0	1811	188.20	1.30	1.1	9.54	0.77	9019.1	43.30
3	74	348	201	537	74.41	0	1.2	0.67	0.86	580.4	17.23
4	74	370	0	584	77.52	1.39	1.3	0.74	0.83	589.7	4.160
5	74	350	204	632	87.08	0	1.4	0.79	0.99	581.9	4.67
10	74	379	214	1750	193.68	0	1.2	0.72	0.82	9491.3	10.39
20	74	349	192	761	91.74	0	1.4	0.82	0.93	587.3	4.92
30	74	372	0	2281	244.48	3.95	2.4	21.32	1.68	8911.5	13.12
100	74	379	0	2034	221.11	2.67	1.0	9.0	0.70	9020.4	11.86
1K	78	346	0	701	102.18	2.56	2.4	1.36	1.49	588.8	5.33
10K	74	348	201	537	74.41	0	1.2	0.67	0.86	580.4	3.99

Average of Aggregate Report

B 4 Aggregate Report of Run the App for validation 2

Users	Median	90% line	95% line	99% line
1	392	453	471	503
2	362	449	491	550
3	366	430	442	499
4	384	438	447	518
5	346	443	459	628
10	357	471	519	811
20	366	428	450	566
30	361	448	487	573
100	360	469	489	710
1K	337	443	479	529
10K	366	430	442	499

Appendix C

Test Case with network A, 4 background applications & Login Average of Summary Report

C 1 Summary Report of Login function for validation 3

users	Samples	Avg	Min	Max	St Dev	Error %	Throughput	Received	sent	Avg bytes	CI
1	1	2221	2221	2221	-	0	27	278.26	0.18	632846.0	-
2	4	447	0	1790	775.09	75	2.2	346.48	0.22	158772.5	285.74
3	1	1824	1824	1824	-	0	32.9	338.82	0.21	632846.0	-
4	1	1887	1887	1887	-	0	31.8	327.51	0.21	632846.0	-
5	1	1740	1740	1740	-	0	34.5	355.18	0.23	632846.0	-
10	1	1797	1797	1797	-	0	33.4	343.91	0.22	632846.0	-
20	2	971	0	1943	971.50	50	1	318.45	0.20	316797	2022.24
30	1	2051	2051	2051	-	0	29.3	301.32	0.19	632846.0	-
100	1	1762	1762	1762	-	0	34.1	350.75	0.22	632846.0	-
1K	2	1019	0	2038	1019	50	58.9	303.6	0.19	316797	2121.12
10K	1	1970	1970	1970	-	0	30.5	313.71	0.20	632846.0	-

Average of Aggregate Report

C 2 Aggregate Report of Login function for validation 3

Users	Median	90% line	95% line	99% line
1	2221	2221	2221	2221
2	0	1790	1790	1790
3	1824	1824	1824	1824
4	1887	1887	1887	1887
5	1740	1740	1740	1740
10	1797	1797	1797	1797
20	0	1943	1943	1943
30	2051	2051	2051	2051
100	1762	1762	1762	1762
1K	0	2038	2038	2038
10K	1970	1970	1970	1970

Test Case with network A, 4 background applications & Run the application

Average of Summary Report

C 3 Summary Report of Run the App for validation 3

users	samples	Avg	Min	Max	St Dev	Error %	Throughput (sec)	Received (kb/sec)	sent (kb/sec)	Avg bytes	CI
1	72	223	0	865	143.81	20.83	2.1	1.29	1.14	616.3	33.79
2	89	315	175	1876	216.13	0	1.3	13.83	0.90	11272.9	45.52
3	57	286	173	415	59.71	0	1.7	0.99	1.17	580.9	15.84
4	74	270	0	1825	209.67	10	1.2	10.35	0.73	9133.0	48.57
5	74	313	167	1900	197.26	0	57.1	8.46	0.66	9110.3	45.70
10	74	291	178	733	75.67	0	1.3	0.74	0.85	585.0	17.53
20	74	313	167	1932	194.71	0	1.1	9.14	0.74	8779.3	45.11
30	78	272	0	401	81.29	3.85	1.4	0.79	0.86	592.7	18.32
100	73	315	0	1789	198.92	5.48	1.6	14.86	1.09	9239.1	46.41
1K	72	291	0	554	77.15	1.39	1.5	0.84	0.95	589.7	18.12
10K	74	320	0	1821	192.38	1.35	2.0	18.16	1.41	9114.9	44.57

Average of Aggregate Report

C 4 Aggregate Report of Run the App for validation 3

Users	Median	90% line	95% line	99% line
1	245	339	345	413
2	286	364	448	467
3	295	354	367	404
4	273	350	366	420
5	294	386	409	467
10	300	349	362	392
20	303	359	390	417
30	279	358	368	398
100	314	378	391	565
1K	298	375	390	424
10K	302	363	378	662

Appendix D

Test Case with network B, 4 background applications & Login Average of Summary Report

D 1 Summary Report of Login function for validation 4

users	samples	Avg	Min	Max	St Dev	Error %	Throughput (sec)	Received (kb/sec)	sent (kb/sec)	Avg bytes	CI
1	1	2221	2221	2221	-	0	27	278.26	0.18	632846.0	-
2	4	447	0	1790	775.09	75	2.2	346.48	0.22	158772.5	179.57
3	1	1824	1824	1824	-	0	32.9	338.82	0.21	632846.0	-
4	1	1887	1887	1887	-	0	31.8	327.51	0.21	632846.0	-
5	1	1740	1740	1740	-	0	34.5	355.18	0.23	632846.0	-
10	1	1797	1797	1797	-	0	33.4	343.91	0.22	632846.0	-
20	2	971	0	1943	971.50	50	1	318.45	0.20	316797	8728.58
30	1	2051	2051	2051	-	0	29.3	301.32	0.19	632846.0	-
100	1	1762	1762	1762	-	0	34.1	350.75	0.22	632846.0	-
1K	2	1019	0	2038	1019	50	58.9	303.6	0.19	316797	9155.35
10K	1	1970	1970	1970	-	0	30.5	313.71	0.20	632846.0	-

Average of Aggregate Report

D 2 Aggregate Report of Login function for validation 2

Users	Median	90% line	95% line	99% line
1	2221	2221	2221	2221
2	0	1790	1790	1790
3	1824	1824	1824	1824
4	1887	1887	1887	1887
5	1740	1740	1740	1740
10	1797	1797	1797	1797
20	0	1943	1943	1943
30	2051	2051	2051	2051
100	1762	1762	1762	1762
1K	0	2038	2038	2038
10K	1970	1970	1970	1970

Test Case with network B, 4 background applications & Run the application Average of Summary Report

D 3 Summary Report of Run the App for validation 4

users	samples	Avg	Min	Max	St Dev	Error %	Throughput	Received	sent	Avg bytes	CI
1	74	400	185	2074	289.65	0	17.01	2.53	0.24	9139.4	67.10
2	74	356	193	1881	247.76	0	43.6	6.39	0.52	9008.9	57.40
3	74	342	192	506	75.5	0	24.8	0.24	0.31	587.6	17.49
4	74	345	192	599	96.17	0	1.1	0.65	0.81	580.6	22.28
5	74	344	0	1537	158.61	1.35	1.1	9.45	0.75	9125.8	36.74
10	74	363	194	1981	204.52	0	55.8	8.18	0.66	9009.4	47.38
20	74	346	0	2369	251.35	1.39	46.9	7.14	0.55	9363.6	58.23
30	74	346	0	614	84.25	1.2	1.1	0.62	0.76	584.1	19.51
100	74	368	202	1741	164.11	0	52.1	6.59	0.61	7761.3	38.02
1K	76	370	0	1718	176.17	1.32	1	8.96	0.73	8905.9	40.12
10K	74	373	0	2126	223.4	1.39	55.2	8.42	0.66	9365.0	51.75

Average of Aggregate Report

D 4 Aggregate Report of Run the App for validation 4

Users	Median	90% line	95% line	99% line
1	361	433	460	1573
2	333	415	443	1368
3	345	423	433	506
4	349	405	449	511
5	348	407	431	455
10	354	416	435	746
20	326	403	412	480
30	357	425	471	506
100	352	457	501	540
1K	376	429	454	560
10K	364	436	458	555

SCRIPT FOR AUTOMATED TESTING OF WEB APPLICATION

Below is the script for automated testing of a web applicaton.

```
Package edmodo.selenium.webdriver;

import java.util.GregorianCalender;

import java.util.conscurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class Edmodo

  {
                WebDriver driver;
                int hour, minute, second, millisecond;
                double current_time, recent_time = 0.0;

                public void CurrentTime(){
                        GregorianCalendar time = new GregorianCalendar();
                        hour = time.get(Calendar.HOUR);
```

```
		minute = time.get(Calendar.MINUTE);
		second = time.get(Calendar.SECOND);
		//System.out.println("Current time is :"+hour+":"+minute+":"+second+":"+millisecond);
		current_time = hour*3600+minute*60+second;
		System.out.println(current_time+";"+(current_time-recent_time));
		recent_time = current_time;
	}


	public void invokeBrowser(){
		try {
			///1
			CurrentTime();
			System.setProperty("webdriver.chrome.driver",
"C:\\Users\\Krish\\Documents\\selenium\\chromedriver_win32\\chromedriver.exe");
			driver = new ChromeDriver();
			driver.manage().deleteAllCookies();
			driver.manage().window().maximize();
			driver.manage().timeouts().implicitlyWait(100, TimeUnit.SECONDS);
			driver.manage().timeouts().pageLoadTimeout(100, TimeUnit.SECONDS);
		   EdmodoTesting();
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

	}
	public void EdmodoTesting(){
		try {
			///2
			CurrentTime();
			driver.get("https://www.edmodo.com/");

			CurrentTime();

			driver.findElement(By.id("qa-test-top-login-button")).click();
			///4
			CurrentTime();
			driver.findElement(By.id("un")).sendKeys("*********");
			driver.findElement(By.id("pw")).sendKeys("********");
			driver.findElement(By.id("qa-test-lightbox-login")).click();
			///5
			CurrentTime();
			driver.findElement(By.xpath(".//*[@id='mainnav']/li[5]")).click();
			driver.findElement(By.xpath(".//*[@id='chats-landing-
page']/div/div[2]/div/div/div/div/div/footer/div/div/form/textarea")).sendKeys("Did you submitted your
Homework??");
```

```
                        driver.findElement(By.xpath(".//*[@id='chats-landing-
page']/div/div[2]/div/div/div/div/div/footer/div/div/form/div[4]/div/button")).click();
                        driver.navigate().back();
                        driver.navigate().back();
                        ///6
                        CurrentTime();

        driver.findElement(By.xpath(".//*[@id='content']/div/div[2]/div[1]/div[3]/div/div[3]/div/div[2]/div[1]/di
v/div/div[6]/div/div[2]/table/tbody/tr/td[2]/div/div/div[1]/textarea")).sendKeys("This is new Assignment");

        driver.findElement(By.xpath(".//*[@id='content']/div/div[2]/div[1]/div[3]/div/div[3]/div/div[2]/div[1]/di
v/div/div[6]/div/div[2]/table/tbody/tr/td[2]/div/div/div[3]/div[2]/button")).click();
                        ///7
                        CurrentTime();

        driver.findElement(By.xpath(".//html/body/div[4]/div/div/div[2]/ul/li[7]/a[1]/span/i")).click();

        driver.findElement(By.xpath(".//html/body/div[4]/div/div/div[2]/ul/li[7]/div/ul/li[9]/a/span")).click();
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                Edmodo myObj = new Edmodo();
                myObj.invokeBrowser();




        }

}
```