Examining Various Input Patterns Effecting Software Application Performance

A Quasi experiment on Performance Testing

Shiva Bhavani Reddy Charla

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden
This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Masters in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Shiva Bhavani Reddy Charla
E-mail: shch15@student.bth.se

External Supervisor
Mr. Srinivas Rao
Test Manager,
Capgemini, Hyderabad.

University Advisor:
Bogdan Marculescu
Doctoral Student
Department of Software Engineering

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

Context Nowadays, non-functional testing has a great impact on the real-time environment. Non-functional testing helps to analyze the performance of the application on both server and client. Load testing attempts to cause the system under test to respond incorrectly in a situation that differs from its normal operation, but rarely encountered in real world use. Examples include providing abnormal inputs to the software or placing real-time software under unexpectedly high loads. High loads are induced over the application to test the performance, but there is a possibility that particular pattern of the low load could also induce load on a real-time system. For example, repeatedly making a request to the system every 11 seconds might cause a fault if the system transitions to standby state after 10 seconds of inactivity.

Objectives The primary aim of this study is to find out various low load input patterns affecting the software, rather than simply high load inputs. A quasi-experiment was chosen as a research method for this study. Performance testing was performed on the web application with the help of a tool called HP load runner. A comparison was made between low load and high load patterns to analyze the performance of the application and to identify bottlenecks under different load.

Methods Performance testing was implemented on a banking application and various input patterns are with the help of a tool called HP load runner. This tool helps in creating virtual users and generating load over the particular operation. The data of input patterns and workload distribution for the application was collected from the interviews conducted with testers and engineers who have domain knowledge about load testing. Metrics used for evaluating the performance of the application are mentioned in the companies SLA. The choice of metrics for this study was considered from the data collected from the domain experts, test engineers and test manager through the interview.

Results The efficiency and productivity of the banking system are affected by low and high input patterns. The comparison was made between low and high input load patterns and also in between various low load patterns based on the throughput, transactional response time and hits per second.

Conclusion The results obtained from the quasi-experiment has shown that some of the low load input patterns are affecting the application performance. The impact of low load patterns is not equivalent to stress caused by the high load on the application, but bottlenecks are identified while implementing low load patterns which are affecting the system performance. Considering low input patterns may enhance the performance of the application and help in identifying bottlenecks. With the help of low load patterns, load testing can be done efficiently by reducing cost and time of the performance tester. This benefits the company to deliver highly stable and reliable product to the client on time with low cost.

Keywords: Nonfunctional testing, Performance testing, Banking Application, Load Testing.
Acknowledgment

I would like to thank my supervisor Bogdan Marculescu, for immense trust and support given to me throughout the process. He has been more like a friend to me rather than a supervisor guiding me through right path and helped me by giving his valuable time and reviews over the research work.

I would also like to thank my external supervisor Mr. Srinivas Rao, Head of our testing team at Capgemini company and my co-employees for helping me in this project. We all have been working as a team while testing this application and they also participated in my research work by giving their valuable time and information regarding the work. His guidance and the team support have led me here to present this report successfully.

I would like to convey my gratitude to the Head of the Software Engineering Department Prof. Jürgen Börstler, for his approval in this research work and giving me an opportunity to present it. And lastly to all my family and friends who has been stuck with me through thick and thin in my research work. Support from my parents and advice from friends has made me this today. Thank you once again.
List of Tables

Table 1: Test pattern data collected from the interviews ................................................. 23
Table 2: Pre-specified condition in SLA ............................................................................. 25
Table 3: High and low load pattern one comparison ......................................................... 30
Table 4: High and low load pattern two comparison ......................................................... 30
Table 5: High and low load pattern three comparison ....................................................... 30
Table 6: High and low load pattern four comparison ......................................................... 30
Table 7: High and low load pattern five comparison ......................................................... 30
Table 8: High and low load pattern six comparison .......................................................... 31
Table 9: Hits per second of all low load patterns .............................................................. 31
Table 10: Throughput of all Low load patterns ................................................................. 32
Table 11: Average transactional response time for high and all low load patterns .......... 32
Table 12: Hits per second of high load and low load pattern 1 .......................................... 34
Table 13: Hits per second of high load and low load pattern 5 ......................................... 35
Table 14: Throughput of high load and low load pattern 1 .............................................. 36
Table 15: Throughput of high load and low load pattern 5 .............................................. 37
Table 16: Average transaction response time of high load and low load pattern 1 ........... 37
Table 17: Average transaction response time of high load and low load pattern 5 .......... 38
Table 18: Comparison of Hits per second of all low load patterns ................................... 38
Table 19: Comparison of Throughput of all low load patterns ....................................... 39
Table 20: Comparison of Average transaction response time of all low load patterns .... 39
Table 21: Occurrence of bottlenecks ................................................................................ 42
List of Figures

Figure 1: Functionality of load runner. [31] ................................................................. 16
Figure 2: Quasi experiment procedure [18] ................................................................. 19
Figure 3: High load testing Running Vuser and hits per second ............................... 27
Figure 4: High load testing Running Vuser and throughput ..................................... 27
Figure 5: Low load testing Running Vuser vs. hits/sec graph ................................... 29
Figure 6: Low load testing Running Vuser vs. throughput bytes/sec graph ............. 29
Figure 7: Hits per second of high load and low load pattern 1 ................................. 34
Figure 8: Hits per second of high load and low load pattern 5 ............................... 35
Figure 9: Throughput of high load and low load pattern 1 .................................... 36
Figure 10: Throughput of high load and low load pattern 5 .................................... 37
Figure 11: Presence of bottlenecks (Test Pattern 1) ................................................ 40
Figure 12: Absence of bottle neck (Load pattern 2) ................................................ 41
Figure 13: Presence of bottleneck (Low load pattern 5) .......................................... 41
Figure 14: Test setting for dynamic session id’s ....................................................... 51
Figure 15: Test setting for dynamic session id’s ....................................................... 52
Figure 16: Graph of high load input hits per second .............................................. 53
Figure 17: Graph of high load input throughput .................................................... 54
Figure 18: Graph of low load pattern one hits per second .................................. 55
Figure 19: Graph of low load pattern one throughput ........................................... 55
Figure 20: Graph of low load pattern one hits per second and throughput ........... 56
Figure 21: Graph of low load pattern two hits per second .................................. 56
Figure 22: Graph of low load pattern two throughput ........................................ 56
Figure 23: Graph of low load pattern two hits per second and throughput ........... 57
Figure 24: Graph of low load pattern three hits per second ................................ 57
Figure 25: Graph of low load pattern three throughput ....................................... 57
Figure 26: Graph of low load pattern three hits per second and throughput .......... 58
Figure 27: Graph of low load pattern four hits per second .................................. 58
Figure 28: Graph of low load pattern four throughput .......................................... 58
Figure 29: Graph of low load pattern four hits per second and throughput .......... 59
Figure 30: Graph of low load pattern five hits per second .................................. 59
Figure 31: Graph of low load pattern five throughput .......................................... 59
Figure 32: Graph of low load pattern five hits per second and throughput .......... 60
Figure 33: Graph of low load pattern six hits per second .................................. 60
Figure 34: Graph of low load pattern six throughput ......................................... 61
Figure 35: Graph of low load pattern 6 hits per second and throughput .......... 61
CONTENTS

ABSTRACT ................................................................................................................................. 1

ACKNOWLEDGMENT .................................................................................................................. 2

LIST OF TABLES .......................................................................................................................... 3

LIST OF FIGURES ........................................................................................................................ 4

1 INTRODUCTION ...................................................................................................................... 7

1.1 BACKGROUND STUDY ....................................................................................................... 8
  1.1.1 Testing ............................................................................................................................ 9
  1.1.2 Non-functional testing .................................................................................................. 9
  1.1.3 Performance testing ..................................................................................................... 9
  1.1.4 Service Level Agreement (SLA): ................................................................................ 10
  1.1.5 Patterns ....................................................................................................................... 10

1.2 RESEARCH GAP ................................................................................................................ 11

1.3 AIM AND OBJECTIVE ...................................................................................................... 12

1.4 RESEARCH QUESTIONS .................................................................................................... 12

2 RELATED WORK .................................................................................................................. 14

2.1 LITERATURE REVIEW ...................................................................................................... 14
  2.1.1 Performance testing .................................................................................................... 14
  2.1.2 Load testing ................................................................................................................ 14
  2.1.3 Stress testing .............................................................................................................. 15

2.2 HP LOAD RUNNER TOOL ................................................................................................. 15

3 METHODOLOGY .................................................................................................................. 17

3.1 RESEARCH METHODS ..................................................................................................... 17
  3.1.1 Survey .......................................................................................................................... 17
  3.1.2 Case study ................................................................................................................... 17
  3.1.3 Experiment .................................................................................................................. 18
  3.1.4 Quasi-experiment ...................................................................................................... 18

3.2 QUASI-EXPERIMENT PROCEDURE .............................................................................. 19
  3.2.1 Banking application .................................................................................................... 20
  3.2.2 Tool functionality ....................................................................................................... 20
  3.2.3 Test settings .............................................................................................................. 20

3.3 VARIABLES ...................................................................................................................... 20
  3.3.1 Independent variables ............................................................................................... 21
  3.3.2 Dependent variables ................................................................................................. 21

3.4 PLANNING AND DESIGN ................................................................................................. 22
  3.4.1 Load testing of high load inputs ................................................................................ 22
  3.4.2 Load testing of low load inputs ................................................................................ 22
  3.4.3 Semi-Structured Interviews ...................................................................................... 22
  3.4.4 List of Test Patterns ................................................................................................. 23

3.5 EXECUTION ....................................................................................................................... 24
  3.5.1 User login to bank application ................................................................................ 24

5
3.5.2 Preconditions for load testing ................................................................. 24
4RESULTS ............................................................................................................ 26
  4.1 HIGH LOAD TESTING ............................................................................. 26
  4.1.1 Running Vusers vs. Hits per sec ....................................................... 26
  4.1.2 Running Vuser vs. Throughput ........................................................ 27
  4.2 LOW LOAD TESTING ............................................................................ 27
  4.2.1 Running Vuser vs. Hits per second .................................................. 28
  4.2.2 Running Vuser vs. throughput ......................................................... 29
  4.3 HIGH AND LOW LOAD PATTERNS ..................................................... 29
  4.4 LOW LOAD PATTERNS ........................................................................ 31
  4.4.1 Hits per second .............................................................................. 31
  4.4.2 Throughput .................................................................................... 31
  4.4.3 Transactional response time ............................................................ 32
5 ANALYSIS ...................................................................................................... 33
  5.1 LOAD COMPARISON BETWEEN HIGH AND LOW LOAD PATTERNS ....... 33
  5.1.1 Hits per second .............................................................................. 33
  5.1.2 Throughput .................................................................................... 35
  5.1.3 Transactional response time ............................................................ 37
  5.2 COMPARISON OF LOW LOADS PATTERNS ...................................... 38
  5.2.1 Hits per second of all low load patterns ........................................ 38
  5.2.2 Throughput of all low load patterns ............................................... 38
  5.2.3 Response time of all low load patterns .......................................... 39
  5.3 BOTTLENECKS IN THE WEB APPLICATION ...................................... 39
  5.4 ANSWERING RESEARCH QUESTIONS ............................................. 42
6 DISCUSSION .................................................................................................. 44
  6.1 THREATS TO VALIDITY ....................................................................... 45
  6.1.1 Internal validity ............................................................................. 45
  6.1.2 External validity ............................................................................. 45
7 CONCLUSION ................................................................................................ 46
  7.1 CONTRIBUTION ................................................................................... 46
  7.2 FUTURE WORK ....................................................................................... 47
REFERENCES .................................................................................................... 48
1 Introduction

Software testing plays an important role in software development life cycle [1]. Verification and validation of product developed can be done by software testing [2]. Many researchers have proposed various methods and techniques to overcome the problems in functional and non-functional testing [1], [3], [4], [5]. Testing helps in assuring the quality of service of the product developed by the software companies and contributed to know to what extent is the application fault tolerant [6]. There is still ongoing research on testing techniques in different fields of software testing.

Software testing can be classified mainly into two types, functional and nonfunctional testing. Functional testing is all about testing the functionalities of the software application. Fulfilling user and technical requirement of the application comes under functional testing. Functional testing checks whether the application is error free, if it responds to user actions such as clicking a button [7].

Nonfunctional testing is analyzing the applications performance under load by calculating response time and overall transaction time for completing the user actions [4]. These all come under nonfunctional requirements of the application, which are mostly neglected by the companies. Non-functional testing is done based on the customer at the end of the testing phase, which is after functional testing. The Company needs to invest more effort, time and cost over nonfunctional testing. Nonfunctional testing is performed based on client’s budget and requirements [8].

Most of the software applications are a web-based application that runs on Internet Browser [9]. The economic importance of web based application increases based on the quality and service provided to the customer. These days non-functional testing has got more significance in real time environment because of its fault tolerance and load handling attributes [10]. One of those nonfunctional testing is performance testing.

Performance testing helps in controlling and improving the software products quality of service. One can determine the application performance based on several factors such as the absence of bottlenecks, low response time, CPU utilization, memory heaps [11]. Performance testing is mainly checking applications threshold or applying an accurate load to evaluate its performance under high load. End to end performance testing is done, before the product gets released to the market [12].

In this research, the author describes various input patterns affecting the performance of the application. Web-based banking applications are being tested by a performance testing tool called Hewlett Packard (HP) load runner which records the scenarios in the form of scripts. This tool was chosen because of ease of integration, weighed against the cost and performance [13].
The main aim of the research is to study the performance of the web application which is subjected to high and low load patterns. The performance and productivity of the web application can be enhanced by analyzing different load patterns [14]. The performance of the system can be affected by inducing load to the web application with a certain number of users accessing a single functionality at a time. We can analyze that either low load or high load inputs are affecting the performance, by studying the behavior of the applications servers [15]. This research work can help the web application developer to enhance the quality and performance of the application by considering the bottlenecks occurred during low load testing. Nonfunctional testing is performed at the ending phase of testing. If the errors are identified at the initial stage of testing, then a lot of time and cost can be saved rather than detecting errors at the end stage [16]. Considering low load patterns will save effort, time and cost for the company. The presence of the bottleneck can affect the performance of the application [17] [18]. Thus identifying bottlenecks with less cost and time by implementing these load patterns will save the company’s investment in testing and also fulfill the customer requirements.

In this report, Section 1 gives an overview of the research work and also describes aim, objective and research gap. Based on the research gap few research questions were framed. Section 2 consist of a literature review of terms defined in the introduction and also gives an overall description of the testing tool and benchmarks. Section 3 is all about methodology, the reason for selecting the quasi-experiment research method and execution was described. Section 4 will have results of the quasi-experiment in the form of tables and graphs. Section 5 describes, how the author has analyzed the results obtained from quasi-experiment. Answering of research questions based on analysis and threats to validity was explained in section 6. Conclusion and future work were mentioned in section 7.

### 1.1 Background study

Software plays a crucial role in complex systems and high-tech application such as financial management, transportation, communication and so on. For assuring the quality and fault tolerance of these complex systems, performance testing is done. Thus non-functional testing has gained more importance in an industrial point of view[12]. There was a time where companies and clients used to prefer more of functional testing rather than non-functional testing. As the number of software users increased and software applications are used widely thereby increasing the attention towards non-functional testing. The main aim of non-functional testing is to enhance the system performance under specific conditions. In this context of the study, the research is mainly based on the performance testing of a web application. The following sections describe software testing and types of testing used in this research.
1.1.1 Testing

Testing is performed to ensure that the product developed is error-free, stable and fault tolerant. These all attributes lead to quality assurance of the product. There are several types of testing, such as regression testing, integration testing, unit testing, functional testing, non-functional testing and so on. Out of these functional and non-functional testing, techniques are dominant and widely used to compare the rest of the techniques[6]. Current research mainly focuses on performance testing, which is one of the non-functional testing techniques. In general, testing fulfills all the functional and non-functional requirements elicited by the customer[2] to develop a bug-free and reliable product. Testing is done to ensure that once the application is deployed to a real time environment, it should be stable and respond as per user actions. Rectifying the product after deployment may cost more for the company than before. Hence testing is done in a such a way that, it should be able to adapt to real-time environment. Hence functional and non-functional testing is performed to make the product more stable and fault tolerant[4], [7].

1.1.2 Non-functional testing

Nonfunctional testing to analyze the performance of applications under load by calculating response time and overall transaction time for completing the user actions [4]. Nonfunctional requirements such as CPU and memory usage, response time, the number of users accessing the application, throughput, memory heaps are some of the metrics used to measure the performance of the complex systems[11]. Non-functional testing is done based on the customer needs and performed at the end of the testing phase. Nonfunctional testing is done using testing tools and these tools are cost, time, effort effective and requires a lot of investment, which is what most of the companies tend to avoid[19]. The Company needs to invest more effort, time and cost over nonfunctional testing. Based on client’s budget and requirements, nonfunctional testing is performed[8]. There are many types of non-functional testing, from which author chose performance testing for his study. The study aims for evaluation of load patterns affecting the software performance, hence performance testing was chosen.

1.1.3 Performance testing

Performance testing is used to measure the end-to-end performance of a system, before going live. Performance testing helps to analyze the systems performance in terms of response time, CPU and memory usage, the number of concurrent users and transactional throughput [11] [14] [20] and simulating real-time scenarios with different load conditions, repeating the load consistently, measuring end user’s response time. System components are monitored under controlled load. Maximum load resisting capacity of an application can be known. Checking application’s stability under expected and unexpected load. The main aspect of performance
testing is for quality assurance and to check the load handling nature when the system is under test. Performance testing is mainly of two types i.e. Stress testing and load testing. Testing the applications threshold limit is stress testing and evaluating the performance of the application under specified load when the system is under testing is load testing [21]. Performance testing mainly consists of two types of testing, stress testing, and load testing.

- **Load Testing**: Load testing is one of the performance testing technique, which is focused on validating the performance characteristics of the application when anticipated workload volumes are implemented [22].
- **Stress Testing**: Stress testing will help in knowing under which conditions the application fails to perform under load i.e., stressing the application of its upper limit, leading to failure in the system [23] [21].

1.1.4 Service Level Agreement (SLA):

SLA is the agreement between the company and customer, where customer requirements are documented and should be achieved by the firm [18]. Therefore, necessary benchmarks for the application are well defined in the SLA. These SLA are entirely confidential, as they consist of the contract agreement between the client and company. SLA is formed based on the TPC-W benchmark standards, where the standard metrics for application and expected performance of the application is calculated through the necessary criteria [24] [25]. TPC-W are standard benchmarks for web servers and database performance defined by the transaction processing performance council[24]. To perform load testing, there have to be few goals set prior to testing. These goals are nothing but benchmarks defined in the respective SLA’s to evaluate the performance of the application based on the metrics. Some of the most commonly used parameters in case of performance testing are response time, throughput hits per second (request sent to the server), CPU utilization and memory heap [18] [26].

1.1.5 Patterns

“A pattern is a recurring solution to a standard problem”[27]. Unexpected pattern of the load from the users which affect the system in the real-time environment is known as load patterns. For knowing the better performance of any system, it is necessary to generate the realistic and predictable patterns of load [22]. In the case of performance testing, workload distribution can also be called as input pattern induced by the application. These distributions are made based on goals specified in the SLA [28].
1.2 Research Gap

In the initial stages of research work a literature review was performed on non-functional testing and performance testing. From the background study, the author has found that there was more research that has been done on functional testing and little research done on nonfunctional testing especially in load patterns [1]. There were very few papers on performance testing and most of them are based on proposing new frameworks and metrics [29]. But there were no papers on various input load patterns and load testing. Thus author is trying to fill the gap by performing a research on evaluating the various load patterns which are affecting the software performance [19] [7]. The input load pattern for this study can be described as a number of users accessing the application for a particular scenario which is defined in the SLA. For example, if the 100 users are accessing the application at the same time performing same user operation such as logging into the application, how the application handle the load and analyze the response time for it. The client sets the benchmarks for these metrics such as response time should be less than 2 seconds, throughput shouldn’t exceed 6 Mbps and so on. These benchmarks act as goals for the performance testing. Thus the workload distribution is made in such a way that, the users are increased gradually from 10 users to 100 users and the result is checked. If the application is able to handle the 100 user load with response time less than 2 second and throughput of 6 Mbps thus the testing was successful and application is able to handle it. If not the reports are sent to development department and asked them to enhance the code and make sure it stable under load. Thus the distribution of users, increasing users step by step is defined as input pattern in this study. If the user load is about 1000 users, then it considered as high load pattern and user load varying from 10-100 users are taken as low load pattern.

For evaluating the proposed load patterns and to estimate its effectiveness over regular high load patterns, an empirical validation was necessary [6]. Quasi-experiment was chosen by the author to conduct his experiment in an industrial environment for comparing the effectiveness of low load patterns over high load patterns on the application. “In a Quasi-experimental design, the program or policy is viewed as an ‘intervention’ in which a treatment – comprising the elements of the program/policy being evaluated – is tested for how well it achieves its objectives, as measured by a pre-specified set of indicators” [30]. The company was using HP Load runner as performance testing tool for testing the banking application. The benchmarks and goals are nothing but the per-specified conditions to be achieved according to the company’s SLA. Thus with access to the company’s data and with the help of a group of software testers from the company this quasi-experiment was performed. By the end of this quasi-experiment author will try to answer the research questions and try to fill the research gap on various load patterns.
1.3 Aim and Objective

**Aim:** The main aim of this research is to identify various input load patterns that are affecting the performance of the system.

**Objective:**
- Conduct performance testing by implementing various load patterns to the web application and observe the performance.
- To observe the response time of all load patterns when the system is under test.
- Analyzing and documenting the reports obtained from different input load patterns and comparing the reports received.
- To identify bottlenecks based on the analysis.
- Interpreting results based on analysis and answer the research questions.

In order to achieve the objectives mentioned above author has conducted his research work in an industrial environment. The web application used for research work is related to an online banking system. This application has been outsourced to the company for high load testing to ensure its performance under load and make it error free. The author was teamed up with a group of testers, who are all assigned for testing this banking application. The objective of the testing has been described in the SLA (Service level agreement). To perform load testing few goals are set before testing. These goals are nothing but benchmarks defined in the respective SLA’s to evaluate the performance of the application based on the metrics. Some of the most commonly used parameters in the case of performance testing are response time, throughput hits per second (request sent to the server), CPU utilization and memory heap [18] [26].

Performance testing was performed with the help of a testing tool called HP load runner. The tools have several functions and can generate Vusers (Virtual Users) which behaves just like real-time user load over the application. According to the SLA, workload distribution was intended to load test this application with a specific amount of users.

1.4 Research questions

**RQ1:** What are the different benchmarks defined for various input load patterns while testing the web application?

**Motivation:** This question was formulated to know the benchmarks defined for input load patterns. These benchmarks help to compare the expected and observed behavior of the load pattern when the application is under load test while performing quasi-experiment.
RQ2: To what extent does low input load pattern effect the performance of the web application compared to standard high input load?

**Motivation:** This question was formulated to know how the low load input patterns affect the behavior of the application. The simple high load is implemented to check if the performance of the application is up to the mark as per defined benchmarks. The maximum throughput and average response time are measured to evaluate the performance and to check whether the application has met user requirements and whether the goals set in SLA.

RQ3: How can bottlenecks be identified, when the application is under load testing?

**Motivation:** Abnormal spikes in the throughput hits per second and delay in response time are an indication of a bottleneck in the application [17] [18]. If there is any, then the performance of the application is affected due to its presence. Identification of the bottlenecks can be made based on the graphs output while performing load testing.
2 Related Work

A literature review has been performed on the nonfunctional testing, web applications, various types of patterns, performance testing tools related to the quasi-experiment. The following sections give a detail explanation of terms defined in the introduction section.

2.1 Literature review

2.1.1 Performance testing

Performance testing is used to measure the end-to-end performance of a system, before going live. It also verifies the systems performance requirements such as response time, the number of concurrent users and transactional throughput [11] [14] [20]. Simulating real time scenarios with different load conditions. These load conditions are like series of actions performed by the users in the real environment, which we define as patterns in this research. The key features of performance testing are

- Repeating the load consistently.
- Measuring end user’s response time.
- System components are monitored under controlled load.
- Application maximum load resisting capacity can be known.
- Checking applications stability under expected and unexpected load.

The main aspect of performance testing is for quality assurance and to check the load handling nature when the system is under test. Performance testing is mainly of two types i.e. Stress testing and load testing. Testing the applications threshold limit is stress testing and evaluating the performance of the application under specified load when the system is under testing is load testing [21].

2.1.2 Load testing

Load testing is one of performance testing, which is focused on validating the performance characteristics of the application when anticipated workload volumes are implemented [22]. These help in detecting functionality errors under load. Load testing evaluated the adequacy of a load balancer. These are the following reasons why we perform load testing on an application[14]

- How many users can application handle before any failure occurs.
- How much data can the database and file server handle?
- Can application handle long periods of usage by multiple users?
- Does the system perform same way across different computers and network configuration, environment, and platforms?
• How will application handle if many users are requesting the certain page at the same time?
• Does system meet its goals for response time, throughput and availability?
• Load testing does identify bottlenecks in the application.

2.1.3 Stress testing

Another type of performance testing is stress testing. There is not much difference between stress and load testing when the system under test is subjected beyond those anticipated workload volumes. Stress testing will help in knowing under which condition the application fail to perform under load. Stressing the application to its upper limit, leading to failure in the system [23] [21]. The benefits of stress testing are

• Determining if data can be corrupted by over stressing the application.
• Assuring that security holes are not opened up under excessive stressful conditions.
• Knowing side effects of common hardware and supporting application failures.
• Estimating how far beyond the target load a request can go before causing any failure and errors.
• Can identify break point of the application.

2.2 HP load runner tool

The tool used for testing the banking application in this quasi-experiment is HP load runner. HP Load Runner tool is a performance testing tool used widely for inducing load or stressing the application. This tool supports all formats of the application irrespective of the code it developed. This tool has been used in most of the companies to study the behavior of client and server side servers [13]. Past literature has also shown that few of researchers have used this tool for testing the banking application [31].

The reason behind using this application for this research is it enables you to test your system under controlled and peak load conditions. The load is generated by running thousands of Virtual users who are distributed all over a network. These virtual users are consistent, repeatable and induce load over the application just like real user with minimum hardware resources[32]. Load runner replaces the human users with the virtual user and emulates an environment of thousands of users working in a client/server system concurrently. This tool records the performance of the application automatically while performing the test. Performance data is presented in the form of graphs and reports. This tool also checks where the performance delay has occurred which helps in improving the performance. As there are automated, we can easily repeat the process more often. Load runner consists of four main components [33]
I. **Virtual user generator (Vugen)**
The first component of Load Runner is Virtual user generator (Vugen). It is used to monitor the communication between the application and server. Generating and recording the script, rendezvous points and transaction are its main functions\[32][33].

II. **Load Controller**
To control all the Vusers in a scenario from single workstation load controller is used. We can also manage and maintain our scenarios. Load controller does distribution of Vuser. The main designing part is done in this component. This component will have two main setting i.e. goal oriented and real-time scenario\[34][35].

III. **Load Generator/Host**
This machine does the execution of Vuser scripts and emulating the actions of a human user by enabling Vusers. Each Vuser is distributed to Load generator by the load controller, when execution the scenario is done\[34][35].

IV. **Analysis**
The last component of load runner tool is Analysis, where we can view performance data in the form of reports and graphs. We can monitor server and network resources through analysis component. During load testing session, Vuser scripts include functions that measure and record the system performance are done in the analysis component\[35][34].

![Figure 1: Functionality of load runner](image-url)
3 Methodology

Based on the research questions framed, quasi-experiment was suitable for this research. The following section gives you a detail explanation of method selection and how the research work was carried out.

3.1 Research methods

This section describes about various research methods [36][37][38]. The advantages and disadvantages of each method and motivation in selecting suitable research method for this study are explained further sections.

3.1.1 Survey

“A survey is a system for collecting information from or about people to describe, compare or explain their knowledge, attitudes, and behavior” [39]. Quantitative and qualitative data gathering can be done through questionnaires and interviews in the survey [40]. Wide data sampling, need more time and response from the target audience. The main aim of this research work is to do a quantitative analysis of the low load patterns. Survey is a formal and structured way of obtaining information [41]. To answer the first research question, the author has performed a semi-structured interviews to gather information about load patterns and their respective benchmarks. Interviews help to gather more appropriate and richer information when compared to other survey methods [19]. Thus author has conducted semi-structured interviews for data collection. After collecting data from the interviews, the author has conducted a quasi-experiment to answer rest of the research questions.

3.1.2 Case study

“Case study in software engineering is an empirical inquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified” [39]. Applicable for studies undefined and explorative in nature. A case study is an observatory study and less controlled when compared to experiment[40]. To isolate the behavior of low load patterns, and focus on them rather than the context [40]. The level of control is less in the case of study when compared to the experiment [39]. There was no ongoing research on the subject and the author has information about testing and metrics to measure the performance, thus there was no need of exploratory side of the case study. This made author not to choose case study as his research method for his research.
3.1.3 Experiment

“Experiment (or controlled experiment) in software engineering is an empirical inquiry that manipulates one factor or variable of the studied setting. Based on randomization, different treatments are applied to or by different subjects, while keeping other variables constant, and measuring the effects on outcome variables. In human-oriented experiments, humans apply different treatments to objects, while in technology-oriented experiments, different technical treatments are applied to different objects.”[39]. Can define the own term and knows expected outcome. Aware of the dependent and independent variable before research. Can repeat the process and observe the variations [40]. There is only one major difference between experiment and quasi-experiment i.e. randomization of data. For this research, the author has predefined conditions defined in SLA, hence experiment was not chosen for this study.

3.1.4 Quasi-experiment

“Quasi-experimental design, the program or policy is viewed as an ‘intervention’ in which a treatment – comprising the elements of the program/policy being evaluated – is tested for how well it achieves its objectives, as measured by a pre-specified set of indicators” [30]. “Quasi-experiment is an empirical inquiry similar to an experiment, where the assignment of treatments to subjects cannot be based on randomization but emerges from the characteristics of the subjects or objects themselves.”[39]. Research questions are solved through quasi-experiment rather than acknowledging from the available literature. The author has complete control and manipulated the conditions i.e. can create a suitable artificial environment for his research work to answer the research questions[42]. There is no randomization of data and pre-specified conditions (Goals set in SLA) are defined in quasi-experiment. For conducting research, the author has conducted a semi-structured interviews for data gathering and implement those patterns in the experiment. The data gathered for the experiment is industrial data, because there was no literature available on specific load patterns and the author has not taken any risk in randomization because of lack of knowledge and data. This made author to conduct his research under an industrial environment where data is valid and experiment will be conducted under domain experts. This condition made author to choose quasi-experiment over rest of research methods to answer rest of the research questions. “A quasi-experiment involves making a change in the value of one variable called the independent variable and observing the effect of that change on another variable called the dependent variable”[36]. Independent variable in this quasi-experiment are various load patterns (running Vusers) and dependent variables are hits per second, throughput and transaction time.
3.2 Quasi-experiment Procedure

From the below figure 2, the quasi-experiment procedure[20] carried away in this manner. First starting with setting up of test environment which includes whole about banking application[43], SLA[18], tool functionality and test setting.

This research design involves the observation of dependent variables by manipulating the independent variables. When the sample size is small and feasible, perform real world setting than true experimental design. The conditions for this design are done by either researcher or domain experts or test engineer or both. Thus the goals of SLA are defined, and pre-specified conditions are available thus this design makes the researcher evaluate the impact of independent variables under naturally occurring conditions [30] [18] [39].

For this research, literature review plays an important role in gathering the information about load testing, performance testing, and nonfunctional testing. A suitable tool is selected to induce load over the open source web application. There might be various types of load affecting the system performance, so here a group of testers who are aware of load distribution is interviewed to gather data regarding various load patterns they are aware of in real time environment. Data collection is done through literature review and interviews. From the given data quasi-experiment is performed by implementing various load patterns to the web application.

This tool can generate Virtual user with its Load generator component. By the help of the tool, we apply the load by defining some users accessing the specific functionality. After applying a load, we should study the behavior by analyzing the data resulting from the quasi-experiment [29]. By increasing the number of users or from low load to high load inputs does the performance is varying or not is observed [44].

![Figure 2: Quasi experiment procedure](image-url)
3.2.1 Banking application

The application used for this research work is an online banking application. According to client requirements, the application will be tested. All the client requirements are documented in SLA, Service Level Agreement. This is an agreement made in between companies[18], where the company has to give a detail report on the performance of the application and check whether the application is fulfilling the client’s requirement or not. HP load runner tool is a suitable tool for testing these type of banking applications [43] and the company was using this tool for testing the application.

3.2.2 Tool functionality

Load runner works on creating virtual users, which are replaced by the real users operating client software. Load generator creates the load on the server by generating many virtual user clients. The controlling of increasing and decreasing of the load is done by controller component i.e. control overload generator. It controls the load test based on scenarios which are compiled scripts. These scripts are designed using Virtual user script generator (Vugen), as it generates the C-language script code to be executed by virtual users by capturing network traffic between applications client and servers. The controller manages each machine status. At the end of each run, the controller merges all the logs obtained from load generator. These logs are sent to analysis component, where these logs are converted into result graphs and reports.

3.2.3 Test settings

While performing this quasi-experiment, there are few changes to be made in the load runner tool. As mentioned above about the Vugen, where the recording of scenarios is done, so that while generating virtual users they behave just like real time user. But while doing so the server doesn’t respond for same session id which has been stored while recording the first scenario. So we need to change the values and enhance the code. For capturing the dynamic values at the server, the response is called correlation. The script modification is done by changing the hard code (refer appendix 1 and 2). For the first iteration you can see the session id, and from vugen we can see the code generated while recording the script, while doing so these code generation code will be copied and pasted over second and further iterations. Thus by this change, we can overcome dynamic values generated by the script and continue with the load distribution in controller component.

3.3 Variables

“A quasi-experiment involves making a change in the value of one variable called the independent variable and observing the effect of that change on another variable called the dependent variable”[36]. Some of the most commonly used metrics in the
case of performance testing are response time, throughput hits per second (request sent to the server), CPU utilization and memory heap [18] [26]. Thus hits per second, throughput and transactional time were considered as dependent variables defined in the SLA. The below section gives you definitions of independent and dependent variables of this experiment.

3.3.1 Independent variables

The variable which is manipulated by the person and doesn’t depend on other variable is known as independent variables. In this scenario, load patterns i.e. number of users accessing are independent variables for this study. One can manipulate the variable with different load patterns i.e. control the flow of user in load testing [39] [45].

3.3.1.1 Load patterns

Total number user workload distribution on the application is known as load patterns. These are also called running Vuser, where behave just like real time user. The number varies on the workload distribution made by the controller. These running virtual users are generated by load generator machine, one of a component of HP load runner tool [46].

3.3.2 Dependent variables

The variable which is affected by the changing the independent variables is known as dependent variables. In this study Hits per second, throughput and transactional response time are affected by the changing the load pattern. With varying number of users accessing the application, the respective metrics are affected. Thus calculation is made based on the values [39] [45].

3.3.2.1 Hits per second

Hits per second depict the total number of hits generated by each and every request. The HPS will follow the same pattern as throughput unless there are any abnormal errors and same page/object are downloaded for a huge number of time.[29]

3.3.2.2 Throughput

Throughput can be defined as a number of transaction or data passed through the server. Once application makes the request, the server sends the data related to the request made by the user. The amount of data transfer is known to be as throughput.[47]

3.3.2.3 Transactional response time

Total time is taken to complete the transaction. The time taken by the server to respond the request made by the application is known as transactional response time. [8]
3.4 Planning and design

3.4.1 Load testing of high load inputs

The basic procedure for performing load testing is to evaluate the performance of the application when a high load is applied to it. To perform high load testing, for this quasi-experiment author has taken 1000 user load, which is known as virtual user load generated by the load generator component as per commands are given by the controller component. These specified user load, metrics, and their expected values are present in the SLA for testing the specific operation. In this experiment author will be testing the user login for the banking application scenario has been taken. Further user operations are explained under the section 3.5.1.

So planning and designing of the high load testing are made by the domain experts or test engineers. The values of SLA are basically concentrated on few of the metrics such as Http request, throughput and transaction time [13]. These values defined in the SLA are based on a calculation made from the standard TPC-W benchmarks [24]. These benchmarks define the standards of performance metrics which are widely used by most of the companies. So SLA is mainly designed based on resources available and measures according to the working environment [25].

3.4.2 Load testing of low load inputs

There has been not much research on various patterns of input load [22]. So this quasi-experiment tries to implement load on the application with various load patterns with a different number of users. The benefits of implementing low load on the application are to save time, cost and effort of the test engineer. Thus author will be performing load testing with low loads, once a certain number of users has induced the effective load. Then further patterns with effective user load are conducted over all design patterns which are collected from the interview. Few goals are to be set prior when we are performing the load testing with the respective scenario.

3.4.3 Semi-Structured Interviews

Semi-structured interviews were conducted for data collection of several load patterns. Due to lack of time and very few employees available for sharing the information about the various test load patterns, the author was able to collect only 10 load patterns. In technical terms, these were also called workload distribution, based on the client and server requirements, the distribution is made [29].

The reason behind conducting semi-structured interviews is to gather data about how workload distribution is done. The people who are working on the same domain and are tester engineers with decent knowledge who has experience in testing was
interviewed. The information given by them are noted down and applied for workload
distribution in testing the application with low input load. As the author was
working with a team for testing the banking web application, they have helped him
in performing this interview successfully.

3.4.4 List of Test Patterns

<table>
<thead>
<tr>
<th>Test pattern</th>
<th>Description of workload distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low load pattern 1</td>
<td>Every ten users access the application in the span of 45 seconds, and when user limit is reached, then the application is under the steady state of 5 minutes' duration. After the steady state is completed, gradually every five users exit the application for every 15 seconds.</td>
</tr>
<tr>
<td>Low load pattern 2</td>
<td>For every 15 seconds, three users are accessed into the application. The duration of steady state is for 10 minutes when defined user limit is achieved. Four users are an exit from the application in span of 10 seconds, once steady state duration has been completed.</td>
</tr>
<tr>
<td>Low load pattern 3</td>
<td>In the span of every 30 seconds, two users are added up to the load in the application. The steady state duration is 3 minutes, then for every 1 minute, one by one user exits from the application.</td>
</tr>
<tr>
<td>Low load pattern 4</td>
<td>Simultaneously all the user limit is added to the application with one-minute duration, and five users leave the application and a steady state for 1 minute, and then five more are leaves the application. After 30 seconds rest of users leaves the application.</td>
</tr>
<tr>
<td>Low load pattern 5</td>
<td>At a time 15 user are added up then after 10 seconds rest of the users are added. The steady state duration is of one-minute. Then gradually all the users are ramped down within the span of 2 seconds.</td>
</tr>
<tr>
<td>Low load pattern 6</td>
<td>For every 10 seconds, the gradual increase of one by one user is raised up until user limit is obtained. After 5 minutes' steady state duration, for every 10 seconds, five users will exit from the application.</td>
</tr>
</tbody>
</table>

Table 1: Test pattern data collected from the interviews.
3.5 Execution

3.5.1 User login to bank application

The main actions performed by the user while accessing this application is logging and signing of the bank application. For this scenario load testing was performed. According to client’s requirement, the application should be able to handle the high load i.e. number of users simultaneously performing the same action. Hence using the HP load runner tool, one real-time user actions are captured, and the same action will be performed by all the virtual users generated by the load generator. With the help of this tool, one can induce load on the application near to the real time scenario.

These are the actions captured by the tool. The user enters the web application’s IP address, and the person is directed to a home page of the banking application. Then selecting the option customer login, one has to enter the user credentials like name and password. Once the credentials are entered, it has to be cross-checked with database regarding the authentication of the user. After successfully logging in to the application one can perform desired actions and has to leave the application without any track.

Thus in this scenarios, whole user actions are recorded by the load generator tool and is being applied across all the virtual users. This recording is converted to the script and as mention in section 3.2.3, test setting dynamic values are correlated. Thus by doing so, the author has applied specified amount virtual users (Independent variable) and observe the dependent variables(metrics). By evaluating these metrics, we can analyze the performance and can also detect any bottlenecks through graphical outputs displayed by the tools while system under test.

3.5.2 Preconditions for load testing

The web application chosen for this research is an online banking system. The economic importance of web based application increases the value of controlling and improving its quality [14] [48] [15]. A quasi-experiment is conducted in an industrial environment using a performance testing tool called HP load runner. The task for the team is to perform load testing on the banking application, where we are given an SLA (Service level document), which are client requirements for the applied load. So we need to satisfy those requirements and report the analysis to the test manager.

Based on the analysis if there are any changes to be made, the issue will be reported to the respective department. Quasi-experiment yields the outcome on various input low load patterns through load testing. Based on benchmarks defined, the comparison between their expected and observed values are made. If there is any difference in observed value other from expected value, then there might be an error in code or
server unable to handle the load. According to the SLA [18] [26] (pre-specified conditions) and information retrieved from the interviews, these threshold limits for following metrics are mentioned in the below table. As per client requirements and from the interviews maximum hits per second, maximum throughput and average transactional response time to be considered for this quasi-experiment. The reason behind selection was as the client wants to know to what extent does the application can handle the load without affecting the performance. When performance testing is performed these are common metrics and their values are considered to evaluate the performance of the application. By comparing the values expected from SLA to observed values in the experiment, one can evaluate the status of application performance for particular scenario [49] [13]. In this case, the author has taken bank login scenario, and these are following expected values for this experiment.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Expected values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High load</td>
</tr>
<tr>
<td>Max Hits per second</td>
<td>&lt; 480</td>
</tr>
<tr>
<td>Max (Bytes/sec) Throughput</td>
<td>&lt;1700000</td>
</tr>
<tr>
<td>Average Transactional response time, (seconds)</td>
<td>&lt; 58</td>
</tr>
</tbody>
</table>

Table 2: Pre-specified condition in SLA
4 Results

4.1 High load testing

According to workload distribution made by the domain experts as per SLA, the high load was being applied to the application. The client requirement was to perform a load test on this banking application of user login scenario. Test engineer has designed this load distribution such a way that, users are added up gradually to observe the stability of the application under test. The gradual increase of running users into the application, controlled by the controller and generated by the load generator. The following sections give you a graphical presentation of results obtained for the respective metrics taken.

4.1.1 Running Vusers vs. Hits per sec

Figure 3 displays the graph between running Vusers (virtual users) and hits per second for 1000 number of user load on the application. The author provides you a graph between running Vuser and hits per second. As running Vusers (virtual users) are increased gradually, even the hits per second are increased. This means as more virtual user access the application, the number of the request are increased and vice versa. When user limit has reached to 1000, a steady state is obtained. The main behavior of the application is observed in the steady state. The graphs show the stability of the application, where it was successful in handling the load of 1000 users.

Thus by considering this graph between running users and hits per second, the author wants to say that application is under control when the specific load applied according to the requirements mentioned in the SLA. The expected value of maximum hits per second should be less than 480 seconds, and observed value is 461.84. These values are considered after performing three iterations. Thus client requirements are met for this metrics and report is submitted to the client regarding the performance of the application under test.
4.1.2 Running Vuser vs. Throughput

From the Figure 4, observe that as running users are increasing, throughput also increased and vice versa. Thus from both the graphs author wants to say that hits per second and throughput are quite similar. Hence the performance of the application is under controlled when the system is under test. The test was conducted within the LAN, there was no major observation regarding the throughput as it was under-utilized for the entire test duration. Further analysis of the graphs is done in the analysis section.

4.2 Low load testing
This section describes all the test patterns taken from the interview and load testing was performed on the application. The following graphs will show the test results of low load testing. Before performing low load testing author has conducted several tests prior with a various number of user load. After effective sessions, the author has found that with user load 50 users, there is the difference in the application performance i.e. observed abnormal spikes in the graphs.

Hence after applying these load patterns to the application most effective are displayed in the below graph, rest of all graphs are in appendices. Author has collected data from 10 interviews. Each input pattern from each interview was noted down and applied with low load (50 Vusers). Out of which after performing experiment 6 of the patterns were most effective and unique behavior had been observed.

These patterns were described in the section 3.4.4 and below graphs gives you a detail explanation of most effective load pattern i.e. load pattern 1 and load pattern 5. From the six unique load patterns, these two patterns have affected the performance of the application significantly. They have exceeded the hits per second, throughput and response time values more than expected. Thus these two were shown and explained in the analysis section. Rest of graphs and their respective values are explained in appendices and further sections. The description of the test pattern are given in the section 3.4.4 and below graphs (Figure 5 and Figure 6) gives you about a detail explanation of load pattern 1.

4.2.1 Running Vuser vs. Hits per second

Figure 5 is the graph between running Vusers (virtual users) and hits per second for the load pattern 1. When user load of 50 users is applied to this application, hits per second has been changing constantly. These changes are shown in form of spikes. When the metrics are compared, observed value of hits per second has exceeded the expected value. Due to difference load pattern (load pattern 1), for every ten users accessing the application in the span of 45 seconds, and when user limit is reached, then the application is under the steady state of 5 minutes’ duration. After the steady state is completed, gradually every five users exit the application for every 15 seconds. This flow of users was implemented in the controller component of the tool and values are observed.
4.2.2 Running Vuser vs. throughput

![Low load testing Running Vuser vs. hits/sec graph](image)

Figure 5: Low load testing Running Vuser vs. hits/sec graph

![Throughput (bytes/sec) graph](image)

Figure 6: Low load testing Running Vuser vs. throughput bytes/sec graph

4.3 High and low load patterns

From the above sections 4.1 and 4.2, we can observe the graphs plotted across running user and hits per second. Microsoft Excel has been used for plotting the graph. For further information regarding the high and low load, testing graphs refer
the appendices. These are following values obtained from performing load testing with three times iterations and frequent value appearing are tabulated.

<table>
<thead>
<tr>
<th>High Load Variables</th>
<th>Low Load pattern 1</th>
<th>Low Load pattern 2</th>
<th>Low Load pattern 3</th>
<th>Low Load pattern 4</th>
<th>Low Load pattern 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Vuser</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
</tr>
<tr>
<td>Hits/sec</td>
<td>42.986, 387.893, 461.844</td>
<td>42.986, 387.893, 461.844</td>
<td>42.986, 387.893, 461.844</td>
<td>42.986, 387.893, 461.844</td>
<td>42.986, 387.893, 461.844</td>
</tr>
<tr>
<td>Throughput Bytes/sec</td>
<td>103961.655, 1369004.413, 1638601.772</td>
<td>103961.655, 1369004.413, 1638601.772</td>
<td>103961.655, 1369004.413, 1638601.772</td>
<td>103961.655, 1369004.413, 1638601.772</td>
<td>103961.655, 1369004.413, 1638601.772</td>
</tr>
</tbody>
</table>

Table 3: High and low load pattern one comparison.

<table>
<thead>
<tr>
<th>High Load Variables</th>
<th>Low Load pattern 2</th>
<th>Low Load pattern 3</th>
<th>Low Load pattern 4</th>
<th>Low Load pattern 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Vuser</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
</tr>
<tr>
<td>Hits/sec</td>
<td>42.986, 387.893, 461.844</td>
<td>42.986, 387.893, 461.844</td>
<td>42.986, 387.893, 461.844</td>
<td>42.986, 387.893, 461.844</td>
</tr>
<tr>
<td>Throughput Bytes/sec</td>
<td>103961.655, 1369004.413, 1638601.772</td>
<td>103961.655, 1369004.413, 1638601.772</td>
<td>103961.655, 1369004.413, 1638601.772</td>
<td>103961.655, 1369004.413, 1638601.772</td>
</tr>
</tbody>
</table>

Table 4: High and low load pattern two comparison.

<table>
<thead>
<tr>
<th>High Load Variables</th>
<th>Low Load pattern 3</th>
<th>Low Load pattern 4</th>
<th>Low Load pattern 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Vuser</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
</tr>
<tr>
<td>Hits/sec</td>
<td>42.986, 387.893, 461.844</td>
<td>42.986, 387.893, 461.844</td>
<td>42.986, 387.893, 461.844</td>
</tr>
<tr>
<td>Throughput Bytes/sec</td>
<td>103961.655, 1369004.413, 1638601.772</td>
<td>103961.655, 1369004.413, 1638601.772</td>
<td>103961.655, 1369004.413, 1638601.772</td>
</tr>
</tbody>
</table>

Table 5: High and low load pattern three comparison.

<table>
<thead>
<tr>
<th>High Load Variables</th>
<th>Low Load pattern 4</th>
<th>Low Load pattern 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Vuser</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
</tr>
<tr>
<td>Hits/sec</td>
<td>42.986, 387.893, 461.844</td>
<td>42.986, 387.893, 461.844</td>
</tr>
<tr>
<td>Throughput Bytes/sec</td>
<td>103961.655, 1369004.413, 1638601.772</td>
<td>103961.655, 1369004.413, 1638601.772</td>
</tr>
</tbody>
</table>

Table 6: High and low load pattern four comparison.

<table>
<thead>
<tr>
<th>High Load Variables</th>
<th>Low Load pattern 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Vuser</td>
<td>Minimum: 0, Average: 526.8, Maximum: 1000</td>
</tr>
<tr>
<td>Hits/sec</td>
<td>42.986, 387.893, 461.844</td>
</tr>
<tr>
<td>Throughput Bytes/sec</td>
<td>103961.655, 1369004.413, 1638601.772</td>
</tr>
</tbody>
</table>

Table 7: High and low load pattern five comparison.
<table>
<thead>
<tr>
<th>High Load Variables</th>
<th>Minimum</th>
<th>Average</th>
<th>Maximum</th>
<th>Low Load pattern 6 Min</th>
<th>Avg</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Vuser</td>
<td>0</td>
<td>526.8</td>
<td>1000</td>
<td>0</td>
<td>24.24</td>
<td>50</td>
</tr>
<tr>
<td>Hits/sec</td>
<td>42.986</td>
<td>387.893</td>
<td>461.844</td>
<td>0.5</td>
<td>9.334</td>
<td>14.813</td>
</tr>
<tr>
<td>Throughput Bytes/sec</td>
<td>103961.655</td>
<td>1369004.413</td>
<td>1638601.772</td>
<td>779.938</td>
<td>12691</td>
<td>20685</td>
</tr>
</tbody>
</table>

Table 8: High and low load pattern six comparison.

4.4 Low load patterns

For the following quasi-experiment, six test patterns have been taken which are collected from the interviews (Table 1). Following tables has values of their respective metrics defined, and these values are obtained after performing three iterations.

4.4.1 Hits per second

Table 9 provides the summary of low load patterns hits per second values and duration of the test when each pattern was applied. Hits per second depict the total number of hits generated by each and every request. The HPS will follow the same pattern as throughput unless there are any abnormal errors and same page/object are downloaded for a huge number of time.[29]

<table>
<thead>
<tr>
<th>Independent variable</th>
<th>Duration of the low load testing with respective test patterns. (mm: ss)</th>
<th>Dependent variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test pattern case number</td>
<td>Number of Vusers</td>
<td>Minimum Hits per second</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>50</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 9: Hits per second of all low load patterns

4.4.2 Throughput

Throughput can be defined as a number of transaction or data passed through the server. Once application makes the request, the server sends the data related to the request made by the user. The amount of data transfer is known to be as throughput [47]. Table 10 provides the information regarding throughput of each pattern applied.
<table>
<thead>
<tr>
<th>Test pattern case number</th>
<th>Number of Vusers</th>
<th>Minimum Throughput Bytes/sec</th>
<th>Average Throughput Bytes/sec</th>
<th>Maximum Throughput Bytes/sec</th>
<th>Median Throughput Bytes/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>0</td>
<td>11889</td>
<td>27317</td>
<td>12050</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>1559</td>
<td>12310</td>
<td>19977</td>
<td>13489</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>888.484</td>
<td>10219</td>
<td>17688</td>
<td>11534</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>0</td>
<td>11667</td>
<td>39153</td>
<td>12874</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>0</td>
<td>11006</td>
<td>39939</td>
<td>7732</td>
</tr>
<tr>
<td>6</td>
<td>50</td>
<td>779.938</td>
<td>12691</td>
<td>20685</td>
<td>15225</td>
</tr>
</tbody>
</table>

Table 10: Throughput of all Low load patterns.

4.4.3 Transactional response time

Table 11 provides the average response time for all the load patterns obtained when the system is under test. Total time is taken to complete the transaction. The time taken by the server to respond the request made by the application is known as transactional response time. [8]

<table>
<thead>
<tr>
<th>Load pattern</th>
<th>Average transactional response time(Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High load</td>
<td>56.33</td>
</tr>
<tr>
<td>Low load pattern 1</td>
<td>2.681</td>
</tr>
<tr>
<td>Low load pattern 2</td>
<td>0.475</td>
</tr>
<tr>
<td>Low load pattern 3</td>
<td>1.045</td>
</tr>
<tr>
<td>Low load pattern 4</td>
<td>6.264</td>
</tr>
<tr>
<td>Low load pattern 5</td>
<td>7.085</td>
</tr>
<tr>
<td>Low load pattern 6</td>
<td>0.261</td>
</tr>
</tbody>
</table>

Table 11: Average transactional response time for high and all low load patterns.
5 Analysis

This section describes the analysis of values received from the quasi-experiment. Performing statistical analysis may show the significance but to what extent does this significance have been affected will be unknown[50]. So, statistical analysis is avoided because the sample data is small and there is a significant difference between the values obtained from the experiment. Thus author has chosen descriptive statistics to provide results in a graphical manner and compare the values to expected values. Descriptive statistics helps in summarizing the data in a systematic way, where readers can understand what exactly author was trying to say through his research work[51] [39]. Descriptive statistics has been chosen for analysis of the data, as conditions are pre-specified in the quasi-experiment and experiment was performed in an industrial environment. The data is collected from valid resource and comparison between various pattern. In this study, the author has taken a high load pattern where the application is stable under load and implemented various low load patterns to the same application to check the stability of the application under low load. The main aim of the descriptive statistics is to observe how the data has been distributed. Comparison between high and low load is performed in section 5.1 to observe the effectiveness of low load on the application when compared to high load[52][53]. Thus, analyzing data with the help of descriptive statistics can help to portray what author want to achieve from his research work[52].

5.1 Load comparison between high and low load patterns

5.1.1 Hits per second

Hits per second depict the total number of hits generated by each and every request. The HPS will follow the same pattern as throughput unless there are any abnormal errors and same page/object are downloaded for a huge number of time [29].

- **Hits per second for high load**: High load input on the application, has attained the highest values as 461.8 hits per second for 1000 users. The observed value is considered to be normal as per benchmark defined the value should be less than 480 hits per second.

- **Hits per second for low load**: When the low load was applied to the application, pattern one, four and five has exceeded value than expected. The load was induced by this application through these specific load pattern. Though the margin is slightly low, it is greater than expected value. The main aim of Load testing is to test the server stability to sustain under low load. Section 5.1.1.1 compares the high load and low load pattern 1 and section 5.1.1.2 compares the high load and low load pattern 5.
5.1.1.1 High load and low load pattern 1

From the graph (Figure 7), the author has compared the hits per second of high load and low load pattern 1. The data points in green circle are data points of low load and rest of points distributed all over the graph are high load data points. When we observe the data points at specific user load, low load patterns haven’t exceeded or overlapped with data points of the high load. The low load pattern 1 was not affecting the performance of the application. Though the values obtained are greater than expected values, when compared to high load these low load pattern wasn’t effective. Table 12 gives expected and observed values. The same is the case with the other low load pattern 5 (section 5.1.1.2). From the observed numerical values, we can assume that pattern is affecting the application performance but when graphically presented there wasn’t any effective load. Thus descriptive statistics has been chosen for representing the data graphically rather than comparing the significance among numerical values.

![Graph: Hits per second of high load and low load pattern 1](image)

**Figure 7: Hits per second of high load and low load pattern 1**

<table>
<thead>
<tr>
<th>Load pattern</th>
<th>Expected value (Hits/sec)</th>
<th>Observed value (Hits/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High load</td>
<td>&lt; 480</td>
<td>461.844</td>
</tr>
<tr>
<td>Low load pattern 1</td>
<td>&lt; 16</td>
<td>18.563</td>
</tr>
</tbody>
</table>

*Table 12: Hits per second of high load and low load pattern 1*
5.1.1.2 High load and low load pattern 5

![Graph: Hits per second vs Running users](image)

**Figure 8: Hits per second of high load and low load pattern 5**

<table>
<thead>
<tr>
<th>Load pattern</th>
<th>Expected value (Hits/sec)</th>
<th>Observed value (Hits/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High load</td>
<td>&lt; 480</td>
<td>461.844</td>
</tr>
<tr>
<td>Low load pattern 5</td>
<td>&lt; 16</td>
<td>25.60</td>
</tr>
</tbody>
</table>

**Table 13: Hits per second of high load and low load pattern 5**

5.1.2 Throughput

Throughput can be defined as a number of transactions or data passed through the server. Once application makes the request, the server sends the data related to the request made by the user. The amount of data transferred is known to be as throughput [47].

- **Throughput for high load:** When the high load is induced on the application, highest throughput observed is 1638601.772 bytes per second. According to the SLA, if the throughput is less than 1700000 bytes per second, the application is in stable condition.
- **Throughput for low load:** When the low load is induced on the application, load pattern one, four and five has exceeded the values than expected. This indicates that load pattern one, four and five are affecting the system performance.

Figure 9 illustrates the graph between throughput of high load and low load pattern 1. Though the observed values have exceeded the expected value, but while comparing the overall performance of application has not been affected. There was
load induced by the low load pattern 1 and has exceeded the expected one. The main aim of Load testing is to test the system stability to hold the load. Maximum throughput is defined in depending variable section. Based on the values, the low load was effective on application performance.

5.1.2.1 High load and low load pattern 1

The graphs of low load pattern 1 in Figure 9 illustrates the data points in the green circle area are of low load pattern and rest of distributed data points are of high load. The numerical values of the both the low load patterns and high load are tabulated in Table 14. Though the numerical values are exceeding, graphically there wasn’t any effective load on the application.

![Throughput bytes/sec vs. Running Vuser](image)

**Figure 9: Throughput of high load and low load pattern 1**

<table>
<thead>
<tr>
<th>Load pattern</th>
<th>Expected value (Throughput bytes/sec)</th>
<th>Observed value (Throughput bytes/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High load</td>
<td>&lt; 1700000</td>
<td>1638601.772</td>
</tr>
<tr>
<td>Low load pattern 1</td>
<td>&lt;25000</td>
<td>27317</td>
</tr>
</tbody>
</table>

**Table 14: Throughput of high load and low load pattern 1**

5.1.2.2 High load and low load pattern 5

The graphs of low load pattern 5 in Figure 10 illustrates the data points in the green circle area are of low load pattern and rest of distributed data points are of high load. Table 15 has the numerical values of the both the low load patterns and high load. Though the numerical values are exceeding, graphically there wasn’t any effective load on the application.
5.1.3 Transactional response time

The time taken by the server to respond the request made by the application is known as transactional response time [8]. The applications transaction response time is 56.33 seconds when the high load was induced on it, which is acceptable because the value is less than expected (58 seconds). When considering the average of all transaction response time of load pattern one, four and five are most effective one. Low load pattern 1 was slightly over the expected but the low load pattern 5 was mere thrice greater than the expected. This behavior is observed when the server is unable to response to the data received while the system is under load. Thus in the case of transaction time, the load is induced on the application by low load pattern one, four and five.

5.1.3.1 High load and low load pattern 1

<table>
<thead>
<tr>
<th>Load pattern</th>
<th>Expected value (Response time) (Seconds)</th>
<th>Observed value (Response time) (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High load</td>
<td>&lt; 58</td>
<td>56.33</td>
</tr>
<tr>
<td>Low load pattern 1</td>
<td>&lt; 2</td>
<td>2.681</td>
</tr>
</tbody>
</table>

Table 16: Average transaction response time of high load and low load pattern 1
5.1.3.2 High load and low load pattern 5

<table>
<thead>
<tr>
<th>Load pattern</th>
<th>Expected value (Response time) (Seconds)</th>
<th>Observed value (Response time) (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High load</td>
<td>&lt; 58</td>
<td>56.33</td>
</tr>
<tr>
<td>Low load pattern 5</td>
<td>&lt; 2</td>
<td>7.085</td>
</tr>
</tbody>
</table>

Table 17: Average transaction response time of high load and low load pattern 5

5.2 Comparison of Low loads patterns

The comparison of six load patterns considered in quasi-experiment is shown in the below tables. The most effective was load pattern one, four and five. Out of these patterns, four and five are relatively similar as you can observe in the appendices. Further explanation of this behavior in between low load patterns is explained in section 5.3.

5.2.1 Hits per second of all low load patterns

The comparison of hits per second of all the load patterns is shown in Table 18. The load pattern one, four and five has more hits per second when compared to the expected value.

<table>
<thead>
<tr>
<th>Load pattern</th>
<th>Expected value (Hits/sec)</th>
<th>Observed value (Hits/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low load pattern 1</td>
<td>&lt; 16</td>
<td>18.563</td>
</tr>
<tr>
<td>Low load pattern 2</td>
<td>&lt; 16</td>
<td>14.438</td>
</tr>
<tr>
<td>Low load pattern 3</td>
<td>&lt; 16</td>
<td>13.078</td>
</tr>
<tr>
<td>Low load pattern 4</td>
<td>&lt; 16</td>
<td>25.06</td>
</tr>
<tr>
<td>Low load pattern 5</td>
<td>&lt; 16</td>
<td>25.60</td>
</tr>
<tr>
<td>Low load pattern 6</td>
<td>&lt; 16</td>
<td>14.813</td>
</tr>
</tbody>
</table>

Table 18: Comparison of Hits per second of all low load patterns.

5.2.2 Throughput of all low load patterns

The comparison of throughput of all the load patterns is shown in Table 19. The load pattern one, four and five has more throughput when compared to the expected value.
<table>
<thead>
<tr>
<th>Load pattern</th>
<th>Expected value (Throughput bytes/sec)</th>
<th>Observed value (Throughput bytes/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low load pattern 1</td>
<td>&lt;25000</td>
<td>27317</td>
</tr>
<tr>
<td>Low load pattern 2</td>
<td>&lt;25000</td>
<td>19977</td>
</tr>
<tr>
<td>Low load pattern 3</td>
<td>&lt;25000</td>
<td>17688</td>
</tr>
<tr>
<td>Low load pattern 4</td>
<td>&lt;25000</td>
<td>39153</td>
</tr>
<tr>
<td>Low load pattern 5</td>
<td>&lt;25000</td>
<td>39939</td>
</tr>
<tr>
<td>Low load pattern 6</td>
<td>&lt;25000</td>
<td>20685</td>
</tr>
</tbody>
</table>

Table 19: Comparison of Throughput of all low load patterns

5.2.3 Response time of all low load patterns

The comparison of average transaction response time of all the load patterns is shown in Table 20. The load pattern one, four and five has response time when compared to the expected value.

<table>
<thead>
<tr>
<th>Load pattern</th>
<th>Expected value (Response time) (Seconds)</th>
<th>Observed value (Response time) (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low load pattern 1</td>
<td>&lt; 2</td>
<td>2.681</td>
</tr>
<tr>
<td>Low load pattern 2</td>
<td>&lt; 2</td>
<td>0.475</td>
</tr>
<tr>
<td>Low load pattern 3</td>
<td>&lt; 2</td>
<td>1.045</td>
</tr>
<tr>
<td>Low load pattern 4</td>
<td>&lt; 2</td>
<td>6.264</td>
</tr>
<tr>
<td>Low load pattern 5</td>
<td>&lt; 2</td>
<td>7.085</td>
</tr>
<tr>
<td>Low load pattern 6</td>
<td>&lt; 2</td>
<td>0.261</td>
</tr>
</tbody>
</table>

Table 20: Comparison of Average transaction response time of all low load patterns.

5.3 Bottlenecks in the web application

A bottleneck is a point in connection situation where the flow of data is impaired or stopped entirely. The spike and sudden drop in the throughput and hits per second lead to higher response time [54] [17]. Effectively there isn’t enough data handling capacity to handle the current volume of traffic. Bottlenecks is a point where data flow has stopped, though the application is under load [17]. There has been no transfer of data, which can be observed from the Figures 11,12 and 13. A bottleneck can occur in the user network or storage fabric or within the server where there is the excessive contention of internal server sources. This stoppage of data slows down and affects the performance especially for database and other heavy transactional applications. In the worst scenario, there is a chance of application crash [17].
From below figures, the presence of bottlenecks is highlighted by the red circle, the points where the data flow has stopped. This stoppage has led to delay in response time, and overall affects the performance of the application. From the numerical values, the pattern 1 and pattern 5 have relatively higher values when compared to the expected values. These abnormal behaviors of those specific patterns have led to the occurrence of bottlenecks in the application. Implementing low load patterns has affected the system performance under load. Low load patterns aren’t effective when compared to high load patterns but bottlenecks are identified when they are compared among other low load patterns. The presence of bottlenecks is known by observing the values from the Table 18, 19 and 20. The load pattern one, four and five have exceeded the expected values. From comparing the values and graphs, bottlenecks are identified. Identifying bottlenecks could be difficult by comparing numerical values so descriptive statistical analysis method is chosen for analyzing the data. Descriptive statistical analysis method was also used for answering the research questions [30], [51], [52].

- Load pattern 1

Figure 11 illustrates the load pattern 1, the metrics throughput and hits per second are overlapped in it. Bottlenecks are identified by observing the abnormal spikes which are highlighted with red circles.

![Figure 11: Presence of bottlenecks (Test Pattern 1)](image-url)
• Load pattern 2

Figure 12 illustrates the load pattern 2, the metrics throughput and hits per second are overlapped. There are no abnormal spikes observed in the graph, thus there are no bottlenecks when load pattern 2 is implemented.

![Figure 12: Absence of bottle neck (Load pattern 2)](image)

• Load pattern 5

Figure 13 illustrates the load pattern 5, the metrics throughput and hits per second are overlapped. The sudden stoppage of data has occurred in the graph, which leads to the presence of bottlenecks when load pattern 5 is implemented.

![Figure 13: Presence of bottleneck (Low load pattern 5)](image)
By referring Table 21, the reader might understand in a better way about the experiment and what author has been trying to say from his research work. Out of 6 patterns, the author has presented a graph of load pattern two where there was no occurrence/presence of bottlenecks.

<table>
<thead>
<tr>
<th>Load patterns</th>
<th>Occurrence of bottlenecks</th>
<th>Refer appendix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load pattern 1</td>
<td>Yes</td>
<td>Figure 20</td>
</tr>
<tr>
<td>Load pattern 2</td>
<td>No</td>
<td>Figure 23</td>
</tr>
<tr>
<td>Load pattern 3</td>
<td>No</td>
<td>Figure 26</td>
</tr>
<tr>
<td>Load pattern 4</td>
<td>Yes</td>
<td>Figure 29</td>
</tr>
<tr>
<td>Load pattern 5</td>
<td>Yes</td>
<td>Figure 32</td>
</tr>
<tr>
<td>Load pattern 6</td>
<td>No</td>
<td>Figure 35</td>
</tr>
</tbody>
</table>

Table 21: Occurrence of bottlenecks

5.4 Answering research questions

RQ1: What are the different benchmarks defined for various input load patterns while testing the web application?

Semi-structured interview is performed by the author to answer this research question. Table 1 is data collected about load patterns from the domain experts and test engineers. Table 2 has pre-specified conditions defined in the SLA for metrics chosen in this research. These tables were the main basis for continuing the further research and perform load testing on a web application. Based on the inputs taken from the interviews which have laid the platform for answering the further research question. Thus answering this research question was main for this research work.

RQ2: To what extent does low input load pattern effect the performance of the web application compared to standard high input load?

From the analysis section 5.1, by comparing maximum hits per second and throughput of high load and low load patterns, there was no effective load on the application by the low load patterns. Though the pattern one, four and five were most effective patterns in order of hits per second or throughput but when compared to high load these all were below the point of high load (Figure 7,8,9 & 10). At the end, the author concludes that when low load patterns are compared to high load, the load was not that effective but when compared to each other patterns, their behavior was different from each pattern.
RQ3: How can bottlenecks be identified, when the application is under load testing?

Abnormal spikes in the throughput hits per second and delay in response time are an indication of a bottleneck in the application [17] [18]. When low load patterns are compared to each other, patterns one, four and five were most effective. In the case of average transaction response time, load pattern five was most effective than rest of the patterns. The value was thrice the expected, which indicates there was a load on the application. Thus bottlenecks are identified under low load testing.
6 Discussion

This section gives the reader a detail explanation of analysis section. The main reason behind doing research on this topic was to find out the effectiveness of low load patterns in load testing. It was challenging for the author in selecting the suitable research method and testing techniques to perform. As quasi-experiment is conducted in an industrial environment, based on valid data taken the analysis has been made.

In this paper, various load patterns are proposed and compared to the benchmarks based on the company domain. The main reason for doing research on low load testing is to check its effectiveness over the performance of the application and finding out any possible bottlenecks occurs while system under low load. There are few situations where the company neglects minor issues which end up costing a fortune in later on stages especially in the testing phase. For saving cost, time and effort, this research was performed.

Making nonfunctional testing more efficient with minimum resource and cost may end up benefitting both the company and the client. Fulfilling customer requirements with least cost and resources is the main goal of the company. This builds trust between company and client which enhances company’s future. Load testing was performed on low load patterns and identifying bottlenecks and any errors affecting the application can be known before applying high load which are more time and cost consuming. The company needs to assign a separate team for load testing the application with the manager and dedicated servers for generating load.

- **Reliability:** Reliability can be defined as to what extent are the results of experiments yield the same values [45]. Author has performed three iterations, for each load pattern and displayed the readings of each metric. The values obtained are similar in all the iterations. From the quasi-experiment results, low load patterns are affecting the performance.

- Though the effect is small, the load was induced to the application due to the presence of bottlenecks. Bottlenecks are quite unexpected while performing low load, but through this quasi-experiment, it has proven that if the application goes under different load patterns rather from the simple high load, performance is affected.

- **Generalizability:** The patterns chosen for load testing are near to real time environment. Author has gathered the information about various load pattern from the domain experts, test engineers who all are aware of real time scenarios. The metrics used for the research are most common used metrics to measure the performance of the application. Banking application used for this research study is a real-time application, due to security and authorization issues author is unable to mention the details of SLA and the banking application name.
6.1 Threats to validity

For any research work, we can observe the presence of threats to validity from the results obtained in the quasi-experiment. [55][41]. This section explains all about threats and the steps taken to mitigate those threats.

6.1.1 Internal validity

The researcher can claim to an extent where independent variables caused the dependent variables.[56]

6.1.1.1 Validity threats

- Only six patterns were considered for the quasi-experiment. While conducting load testing with these patterns, it was observed that most behaviors of the patterns are similar to each other. Thus patterns with unique behaviors were selected.

- The informal interview was performed due to time constraint and work in the office. The procedure for formal interview needs much more time from the testers, which was not possible with limited resources. Hence during breaks and after office hours the interviews was conducted.

6.1.2 External validity

Legitimately generalizing the findings from the research are known as external validity[56].

6.1.2.1 Validity threats

- Benchmarks defined in the SLA can be one of external validity as the quasi-experiment was performed in the industrial environment. Every company has set their goals while performing load testing. These goals are quite similar to most of TPC-W benchmarks standards defined for most of the performance testing. Based on the resources and customer requirements goals may vary, but the benchmarks remain same throughout the testing.

- The results obtained from the quasi-experiment can be generalized as the experiment was conducted in a simulated environment with conditions that are similar to a real time scenario. Therefore, it can be said that the results obtained from the experiment can be generalized to scenarios that are similar to the conditions that were considered in the experiment.

- The tool and metrics used for analyzing the performance of banking application are suggested by the domain experts and widely by most of the researchers [13].
7 Conclusion

The final conclusion of the study is when the low load patterns are compared to high load pattern, they were not that effective but in between low load patterns, there is a vast difference in hits per second, throughput and transactional response time to expected values. Bottlenecks can be identified with the help of the low load patterns. Low load patterns can induce load over the application and occurrence of bottlenecks in the application can be known. Implementing low load patterns can save time, effort and money on testing for the company. This research work can be helpful for further studies who are doing research in various fields such as performance testing, improving the quality of service in web applications, load, and stress patterns. From the analysis section author has concluded that, even with different low load patterns, the load can be induced by the application. The results obtained from the quasi-experiment proves that there is a chance of bottlenecks in a web application which is under low load testing. Thus this experiment is limited and conducted based on the data given by the company (SLA). Every company has their own SLA’s and are designed based on client requirements. Excluding all the limitations, there is a chance of inducing load on the application with various input patterns rather than simply high.

7.1 Contribution

The present research work can provide assistance for performing load testing on banking application with different load patterns. Test case designer can adopt the pattern used in the present study for workload distribution and load testing the application. Performance tester can implement low load patterns to test the application and identify bottlenecks with less effort and time.

For testing banking application, one can use metrics and values for analyzing the performance of it when the system is under load. These days’ companies are investing more time, money and effort on testing, which can be reduced and within given resources testing can be performed. From limited resources and based on the data provided to the researcher, he was successful in testing with various input patterns on the application. This research work might fill few research gaps and can provide further assistance for testing on load patterns and nonfunctional testing.
7.2 Future work

Future work for this study could be testing the open source application with other metrics or with the same metrics. More load patterns can be induced over the application with varying number of user load. Number of concurrent users and other load testing tool can also be used for performance testing. Load patterns can be applied to other non-functional testing. This report can be used for performing load testing on other applications with open source testing tools and other metrics. With limited resources, quasi-experiment was performed and was successful in impending load over it with various load patterns. Much more validation of results is left to the future work. The generalization of research work is based on results obtained from these specific load patterns which were designed based on the company SLA. This might affect the generalization, but the author was able to induce load with these patterns on the application. In future, this study can lay platform for load testing the real time system in large scale and with more number of input load patterns.
References


Figure 14: Test setting for dynamic session id's
Figure 15: Test setting for dynamic session id's
Figure 16: Graph of high load input hits per second
Figure 17: Graph of high load input throughput
Figure 18: Graph of low load pattern one hits per second

Figure 19: Graph of low load pattern one throughput
Figure 20: Graph of low load pattern one hits per second and throughput

Figure 21: Graph of low load pattern two hits per second

Figure 22: Graph of low load pattern two throughput
Figure 23: Graph of low load pattern two hits per second and throughput

Figure 24: Graph of low load pattern three hits per second

Figure 25: Graph of low load pattern three throughput
Figure 26: Graph of low load pattern three hits per second and throughput

Figure 27: Graph of low load pattern four hits per second

Figure 28: Graph of low load pattern four throughput
Figure 29: Graph of low load pattern four hits per second and throughput.

Figure 30: Graph of low load pattern five hits per second.

Figure 31: Graph of low load pattern five throughput.
Figure 32: Graph of low load pattern five hits per second and throughput

Figure 33: Graph of low load pattern six hits per second
Figure 34: Graph of low load pattern six throughput

Figure 35: Graph of low load pattern 6 hits per second and throughput