Weighted Fair Distribution of Web Requests on E-commerce Platforms Using Citrix ADC Load Balancer.

Leslie Lemarian

133481

CNS

Supervisor Name

Tiberius Tabulu

Submitted in Partial Fulfillment of the Requirements of the Bachelor of Science in

Computer Networks and Cybersecurity

School of Computing and Engineering Science

Strathmore University

Nairobi, Kenya

June 2023

# Declaration and Approval

I declare that this work has not been previously submitted and approved for the award of a

degree by this or any other University. To the best of my knowledge and belief, the research

proposal contains no material previously published or written by another person except where due reference is made in the research proposal itself.

Student Name: Leslie Lemarian

Admission Number: 133481

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Approval Statement**

The Proposal of Leslie Lemarian has been reviewed and approved by Mr.Tiberius Tabulu

Supervisor Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

Server overload is a common challenge facing e-commerce businesses today. Server overload can cause website downtime, slow page loading, and other performance-related problems that can cause great customer dissatisfaction and lead to loss of revenue. In this project, there will be load balancing using Citrix ADC load balancer. Citrix ADC is a popular load balancer that provides advanced features for traffic management, security, and analytics.

This project will implement a weighted round-robin algorithm on the load balancer. The weighted round-robin algorithm is a load-balancing algorithm that assigns weights to each server based on its processing power so that the server with more capacity handles more traffic than servers with less capacity. This project will also be monitoring the traffic using Citrix ADC analytics tools to gain insights into traffic patterns. The project aims to demonstrate the effectiveness of load balancing using Citrix ADC and the weighted round-robin algorithm in solving server overload in e-commerce businesses. It will improve website performance and enhance customer experience ultimately leading to increased customer loyalty and revenue growth.

**Table of Contents**

[Declaration and Approval ii](#_Toc136861455)

[Abstract iii](#_Toc136861456)

[List of Figures vi](#_Toc136861457)

[List of Tables vii](#_Toc136861458)

[List of Abbreviations viii](#_Toc136861459)

[Chapter 1. Introduction 1](#_Toc136861460)

[1.1 Background 1](#_Toc136861461)

[1.2 Problem statement 2](#_Toc136861462)

[1.3 Research Objectives 2](#_Toc136861463)

[1.3.1 General aim 2](#_Toc136861464)

[1.3.2 Specific Objectives 2](#_Toc136861465)

[1.4 Research questions 3](#_Toc136861466)

[1.5 Justification 3](#_Toc136861467)

[1.6 Scope 4](#_Toc136861468)

[1.7 Limitations and Delimitations 4](#_Toc136861469)

[Chapter 2. Literature Review 2](#_Toc136861470)

[2.1 Introduction 2](#_Toc136861471)

[2.2 Web Services and Web Requests 2](#_Toc136861472)

[2.3 Load Balancing in Web Services 2](#_Toc136861473)

[2.4 Load balancing Algorithms 3](#_Toc136861474)

[2.5 Related Works 4](#_Toc136861475)

[2.5.1 High availability design 4](#_Toc136861476)

[2.5.2 Ant Colony Optimization 4](#_Toc136861477)

[2.6 Conceptual Framework 5](#_Toc136861478)

[Chapter 3. Methodology 7](#_Toc136861479)

[3.1 Introduction 7](#_Toc136861480)

[3.2 System Development Methodology 7](#_Toc136861481)

[3.3 Analysis 7](#_Toc136861482)

[3.4 Design 8](#_Toc136861483)

[3.5 Code/Configuration 8](#_Toc136861484)

[3.6 Test 8](#_Toc136861485)

[3.7 Deployment 8](#_Toc136861486)

[3.8 Deliverables 9](#_Toc136861487)

[3.9 Tools and Techniques 9](#_Toc136861488)

[3.9.1 Software requirements 9](#_Toc136861489)

[BIBLIOGRAPHY 10](#_Toc136861490)

# List of Figures

[Figure 2.1 Conceptual Framework 6](#_Toc136861434)

[Figure 3.1 Methodology 7](#_Toc136861435)

# List of Tables

[Table 2.1 Difference between algorithms 4](#_Toc136861448)

# List of Abbreviations

WRR-Weighted Round Robin

# Introduction

## Background

Server overload is a common problem among many businesses and organizations where they rely heavily on web applications and services. Due to the presence of a high number of requests in large data centers, using a single server will not be enough to provide service to all incoming requests. Hence 'n' number of servers are used to provide efficiency in attending to incoming requests(Raghul et al., 2017). With the 'n' number of servers, we need to distribute requests among them considering the capacity each server can hold. Different web servers need to be load balanced so that no server would be overloaded or underloaded. The requests thus will have a better response time and less processing time. (Sharma, 2018). Server overload can lead to many issues including; decreased productivity, lost revenue, and reputational damage. In cases where the servers are overloaded, they become vulnerable to cyber-attacks.

Saeed Sharifian et al. (2015) proposed a load-balancing algorithm that classifies requests into different classes. The algorithm dynamically selects a request from a class and assigns the request to a server. For both scheduling and dispatching, new probabilistic algorithms are proposed. To avoid using unreliable measured utilization in the face of fluctuating loads the proposed load-balancing algorithm benefits from a queuing model to predict the utilization of each server. The author also used a control loop feedback to adjust the predicted values periodically based on soft computing techniques (Zongyu & Xingxuan, 2015). The weighted round-robin algorithm is a scheduling algorithm that prioritizes the capacity of the servers and the load is shared depending on its capacity.

Server failures can significantly impact the availability and reliability of e-commerce platforms. The unavailability of e-commerce platforms due to server failures will greatly affect the business. The business will lose revenue because no one can access their e-commerce platform and make the purchases they wanted.

In a system where the end goal is to efficiently distribute incoming requests, and e-commerce platforms, to ensure high availability, load-balancing techniques are highly utilized. Load balancing ensures there’s optimum performance, maximum resource utilization, and high availability.

A weighted round-robin algorithm makes sure that there’s full resource utilization. It achieves that by assigning different weights to the different servers based on their capacity and performance. The weights determine the proportion of requests each server receives. The use of a weighted round-robin algorithm will guarantee resource availability, better e-commerce platform performance, and resource availability.

## Problem statement

In today’s world, there is an increased reliance on web-based applications and services by businesses, organizations, and potential customers. That has brought about server overload as a challenge. Under high traffic load, e-commerce platforms often struggle to efficiently handle incoming requests, leading to degraded performance, increased response times, and even complete outages.

To address the above challenge, this project aims to implement a load-balancing solution using a Citrix ADC load balancer that implements a weighted round-robin algorithm. WRR combines priority scheduling based on resource availability on the available 'n' servers. This will distribute incoming requests across multiple servers according to their capacity thereby improving resource utilization and reducing the probability of downtime as a result of server overloads. Additionally, the response times, availability, and scalability will be significantly improved.

## Research Objectives

### General aim

The aim of this project is the implementation of a load-balancing solution that will enhance the overall performance and responsiveness of web servers considering a distributed e-commerce environment. The algorithm implemented will distribute incoming requests across web servers to ensure a weighted and balanced workload distribution and lead to maximum resource utilization.

### Specific Objectives

1. To investigate load balancing concepts on web services
2. To review load balancing algorithms that can be used on web services.
3. To review related existing works on load-balancing solutions for web services
4. To configure and deploy a weighted round-robin algorithm on a Citrix ADC load balancer
5. To test the configured weighted round-robin load balancing solution

## Research questions

1. What are the load-balancing concepts used in web services and how do they impact performance?
2. What are the different load-balancing algorithms available for web services and how do they compare in terms of effectiveness?
3. What are the existing works and research studies on load-balancing solutions for web services, and what are their findings and recommendations?
4. How can a weighted round-robin algorithm be configured and deployed on a Citrix ADC load balancer for optimizing load distribution in web services?

## Justification

The demand for web-based services continues to grow day by day as many organizations and businesses are moving to provide their services on the web. Server overload characterized by high traffic and resource-intensive requests poses a significant challenge for organizations seeking to deliver good customer service. To address this issue, implementing a load balancer becomes crucial.

The implementation of a citrix adc load balancer with the weighted round-robin algorithm will increase redundancy and make sure that web services are highly available. By intelligently distributing requests, the load balancer ensures that if one server fails, the work is seamlessly shifted to the other functioning servers. In this project, the load balancer will be fully aware of the load on the servers and the remaining capacity so that it can make decisions to allocate the load efficiently(Mohammed et al., 2018).

This project's outcome will greatly improve the performance of the web services. As Sadia et al.,( 2017) discussed weighted round-robin intelligently assigns weights to server resources based on their capabilities allowing more powerful servers to handle a larger share of the load.

## Scope

This project will apply the Weighted Round Robin algorithm for load balancing on an ADC load balancer. This is because distributed web servers have different capabilities and thus could handle a varied number of web requests per unit of time. The testbed will be a virtualized setup that mimics a web server environment.

## Limitations and Delimitations

As the implementation of this load-balancing solution was being implemented, there came a few problems that the researcher faced.

1. Virtual environment constraints- the project will be limited to available memory and storage.
2. The capability of devices- the project is limited to the capabilities of the hypervisor.
3. The version of Citrix ADC server and the supported capabilities.

# Literature Review

## Introduction

This chapter aims to provide a comprehensive overview of load-balancing concepts in web services, exploring underlying principles, review of load-balancing algorithms that are commonly used in web services, and challenges associated with load balancing.

## Web Services and Web Requests

In the current day, businesses providing web services have multiple solutions to try to control traffic. Different businesses have different requirements and choose different techniques and tools based on their needs and budget. There is a wide range of options that a business can choose to try to solve the same problem this project is solving, server overload. Traffic shaping can be used. It involves restricting network traffic based on predefined policies. It allows businesses to control the flow of data and prioritize certain types of traffic over others((Xiao et al., 2017)). A more common one is using reverse proxies. The reverse proxy acts as an intermediary between client devices and web servers. The proxy receives requests on behalf of servers, does load balancing, and forwards the requests to the appropriate servers. There are also web application firewalls that protect web servers. It monitors HTTP traffic and blocks malicious traffic. For this project load balancing was picked because it provides more of what was necessary. A load balancer can perform traffic distribution, load monitoring, server health checks, session persistence,content-based routing, graphic load balancing, scalability, and security features.

## Load Balancing in Web Services

In web services, load balancing is vital to improving turnaround times and user experience. There are different types of load balancers based on the traffic it routes. Application load balancers route HTTP/HTTPS traffic and network load balancer route traffic to a healthy resource. In trying to curb server overload, there has been a lot of development in introducing or furthering the already existing load balancing theories. In case of a server failure, an efficient load-balanced system can automatically redirect traffic to healthy servers, minimizing downtime and ensuring continuous service availability.

Load balancing distributes incoming traffic across multiple servers, creating redundancy and reducing reliance on a single server. Load balancing enhances fault tolerance by detecting server failures and automatically redirecting traffic to healthy servers. By quickly identifying and isolating faulty servers, load balancing ensures that users are seamlessly routed to operational servers, minimizing the impact of failures on overall service availability.

Load balancing techniques can also maintain session persistence, ensuring that requests from the same client are consistently routed to the same server. This is crucial for maintaining session data integrity and providing a seamless user experience. Efficient load balancing mechanisms ensure that session persistence is maintained accurately, preventing disruptions in user sessions and maintaining service availability.

By ensuring efficient resource utilization and intelligent traffic management, load balancing plays a vital role in delivering reliable and highly available web services to users.

## Load balancing Algorithms

Round robin is arguably the simplest to implement and most widely used in web services. Round robin distributes incoming traffic equally among available servers in a cyclic manner. Each server gets its turn to server a request and it repeats. However, as Deepa & Cheelu (2017) opine, this does not take into account the actual load or capacity of each server potentially leading to server inefficient distribution.

A further study came up with a randomized load-balancing algorithm. It tried to reduce biases by distributing requests randomly but to implement it you had to assume that the servers had similar capacities and the workload is relatively uniform. That might sound correct but in web services, it should be noted that different workloads are different. Take an example of a login and a signup request. Login requests should be given higher priority and directed to servers with higher capacity for quick authentication. On the other hand, a sign-up request could be handled by servers with less capacity for storing new user information. To further the above two algorithms, the weighted round robin improves upon the basics of the round robin by assigning different weights to each resource based on their capacity. The weights determine the proportion of requests each resource receives. Servers with higher weights handle more requests while the lower weights handle fewer requests (Manirabona et al., 2016). The subtle differences are highlighted in Table 2.1. This approach allows for better load balancing based on the capabilities of each server.

Table 2.1 Difference between algorithms

|  |  |
| --- | --- |
| **Weighted round-robin** | **Round robin** |
| Assigns requests based on server weights and balances load accordingly | Assigns requests cyclically without considering server weights |
| Considers server capacities and performance | Does not consider server capacity and performance |
| Suitable for scenarios with heterogeneous server capacities | Suitable for scenarios with uniform server capacities and workload |

## Related Works

### High availability design

A lot has been done to try to reduce the gaps and solve the problem, of server overload.

To achieve high availability on e-commerce platforms(Zhao et al., 2015) introduced a high-availability design that made sure even when the worker node failed, migrate tasks from failed nodes to other nodes. The author thought of server overload in its extreme state where even the server itself has already failed. The goal was to achieve availability even when a working server has been overloaded and crashed.

The author achieved what he/she wanted but at the cost of losing a server. They should have thought of saving it before it got that extreme and maybe the server would not have crashed.

### Ant Colony Optimization

This is a proposed technique by (Xiao et al., 2017) that solves the scheduling of tasks. Ant colony optimization gets its inspiration from the natural behavior of ant colonies for searching for food and connecting through pheromone trails that are left behind by ants. Ant colony optimization for task scheduling leverages the collective intelligence of the ant colony to explore and exploit the solution space, gradually converging toward an optimal or near-optimal task-resource assignment. By incorporating the pheromone trail and heuristic information, the algorithm balances exploration and exploitation, allowing for efficient scheduling of tasks while considering various constraints and objectives. Its performance can be sensitive to the problem size, particularly in large-scale task scheduling scenarios. As the number of tasks and resources increases, the complexity of the search space grows exponentially, potentially leading to increased computation time and difficulty in finding optimal solutions within a reasonable timeframe.

## Conceptual Framework

A user accesses an e-commerce platform by entering the websites uniform resource locator.The user’s request is sent to the platform’s load balancer. The load balancer, in this case, acts as the entry point for all incoming requests. The load balancer solution,the citrix adc load balancer with the weighted robin, determines the most appropriate server to handle the request. The weighted round robin algorithm takes into account the predefined weights assigned to each web server. The load balancing solution will pick the server to answer the request based on information it has about the servers; servers weights, current availability and capacity of the servers. The load balancer will forward the users request to the selected server by modifying the destination ip address to the one of the server. The server receives the forwarded request from the load balancer and asks for the required response from the backend database server. Once the server connected directly to the load balancer receives the response from the backend database server,it forwards it to the load balancer. The load balancer receives the response from the local web servers and modifies the response’s source address to appear as if it originated from the load balancer itself. The response is then sent to the user who initially sent the request.

The users device receives and renders the response, displaying the requested content from the e-commerce platform.Throughout that process, the load balancer continuously monitors the availability and performance of web servers.

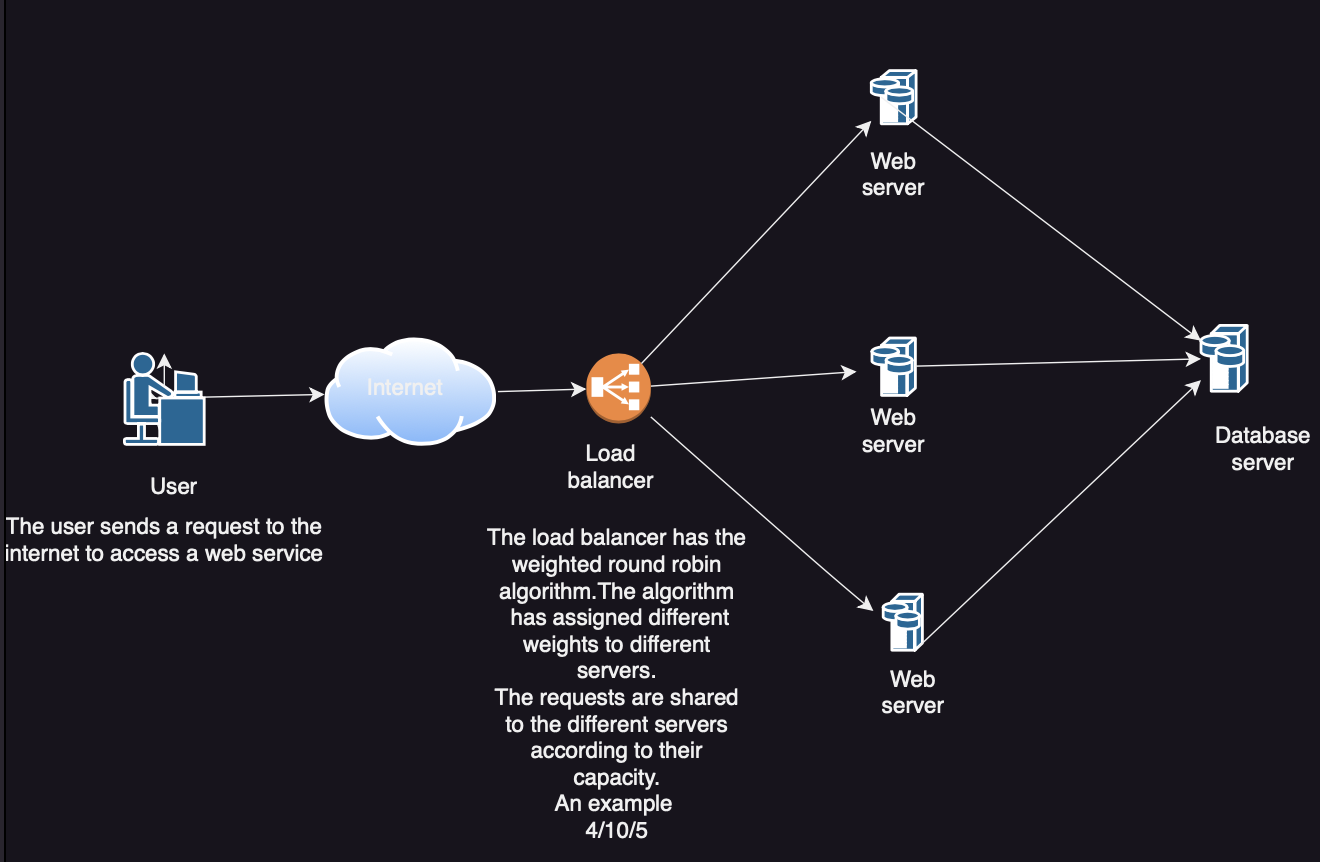


Figure . Conceptual Framework

# Methodology

## Introduction

To address server overload, an incremental methodology approach is utilized. Using this approach the project will be adding new updates and testing from an already existing roadmap.

## System Development Methodology

This project will use an incremental methodology. This approach improves on an already created roadmap. Everything will be added as an update. It will provide a clear and straightforward framework as each stage is well-defined and the process is easy to follow. The picked methodology has the following steps.

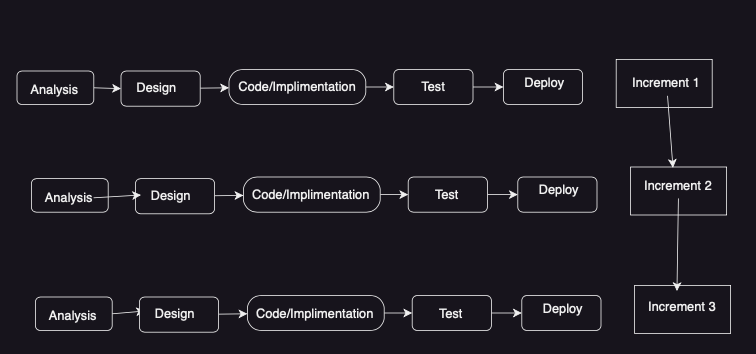


Figure . Methodology

## Analysis

This phase looks at requirements gathering and analysis. This project will consider secondary data from journal articles, papers, and other technical documentation. This data will be considered in the design of the prototype and the primary data gathered from the prototype will be used to measure the efficiency of the prototype.

## Design

A logical design of the proposed testbed is defined in this stage. This is derived from the analysis phase discussed in section 3.2.1. The designs will be represented using the following diagrams:

1. Use case
2. FSM diagram
3. Network architecture/topology
4. Activity diagram

## Code/Configuration

Since the development and project will require a network architecture, it will need virtualization software to create machines virtually, oracle virtual box. There will be a need for three virtual machines to represent the local web servers and an extra virtual machine to represent the backend server. The backend server should be connected to a database, apache mysql. The project will need an IDE like visual studio for modifying configuration files. The project will need load testing tools like Apache Jmeter for monitoring network traffic and performance.

## Test

After the load balancing code has been added to the design, the testing will commence. This phase will greatly help in analyzing vulnerabilities and checking weaknesses. Feedback will be taken from this phase and written as increment 1. The feedback will be very helpful going forward as it will set the tone for increment 2, increment 3, and so forth.

In this, there will be a checking of the allocation of requests against the proposed Weighted Round Robin algorithm.

## Deployment

After solving all vulnerabilities and weaknesses, the load-balancing solution will be deployed into an e-commerce platform.

This is outside the scope of this process, however, detailed documentation on the configuration process and a possible deployment plan can be provided for this project

## Deliverables

1. Proposal
2. Analysis and Design Diagrams
3. Prototype of a solution that has a weighted round-robin algorithm implemented on a Citrix ADC configuration
4. Test Cases
5. Final documentation Report

## Tools and Techniques

### Software requirements

1. Citrix ADC load balancer
2. Virtualbox
3. Apache Jmeter
4. Apache mysql

# BIBLIOGRAPHY

Deepa, T., & Cheelu, D. (2017). A comparative study of static and dynamic load balancing algorithms in cloud computing. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, 3375–3378. https://doi.org/10.1109/ICECDS.2017.8390086

Manirabona, A., Boudjit, S., & Fourati, L. C. (2016). A Priority-Weighted Round Robin scheduling strategy for a WBAN-based healthcare monitoring system. *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 224–229. https://doi.org/10.1109/CCNC.2016.7444760

Mohammed, M. A., Hasan, R. A., Ahmed, M. A., Tapus, N., Shanan, M. A., Khaleel, M. K., & Ali, Ahmed. H. (2018). A Focal load balancer based algorithm for task assignment in cloud environment. *2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 1–4. https://doi.org/10.1109/ECAI.2018.8679043

Raghul, S., Subashri, T., & Vimal, K. R. (2017). Literature survey on traffic-based server load balancing using SDN and open flow. *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, 1–6. https://doi.org/10.1109/ICSCN.2017.8085416

Sadia, F., Jahan, N., Rawshan, L., Jeba, M. T., & Bhuiyan, T. (2017). A priority based dynamic resource mapping algorithm for load balancing in cloud. *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, 176–180. https://doi.org/10.1109/ICAEE.2017.8255349

Sharma, D. (2018). Response Time Based Balancing of Load in Web Server Clusters. *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 471–476. https://doi.org/10.1109/ICRITO.2018.8748373

Xiao, G., Wenjun, W., Jiaming, Z., Chao, F., & Yanhua, Z. (2017). An Openflow Based Dynamic Traffic Scheduling Strategy for Load Balancing. *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, 531–535. https://doi.org/10.1109/CompComm.2017.8322602

Zhao, L., Chuang, Z., & Ke-fu, X. (2015). SpeedStream: A real-time stream data processing platform in the cloud. *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, 1–10. https://doi.org/10.1109/PCCC.2015.7410267

Zongyu, X., & Xingxuan, W. (2015). A predictive modified round robin scheduling algorithm for web server clusters. *2015 34th Chinese Control Conference (CCC)*, 5804–5808. https://doi.org/10.1109/ChiCC.2015.7260546

**Strathmore University**

**School of Computing and Engineering Sciences**

**Project Proposal Assessment Guide**

|  |  |
| --- | --- |
| **Student Number** | **133481** |
| **Working Title:** | Weighted Fair Distribution of Web Requests on E-commerce Platforms Using Citrix ADC Load Balancer |

|  |  |  |  |
| --- | --- | --- | --- |
| **Evaluation Areas** | **Weight** | **Score** | **Notes** |
| **Title page:**  Informative, concise, and appropriate | **2 pts** |  |  |
| **Abstract**  To have background, problem, solution, methodology (approach data and tools) outcomes and expectations | **2 pts** |  |  |
| **Introduction**  Background **(2)**  *A clear illustration of issue, context and audience*  Problem Statement **(2)**  *Pain points, audience, who is affected and how solution comes in to fix the pain.*  Objectives (S.M.A.R.T and Linked to Problem Statement) **(2)**  Research questions **(1)**  *Alignment of questions with objectives*  Justification **(2)**  *Should be research supported.*  Scope of Project **(2)**  *Specify boundaries of people process, HW/SW, data etc.*  Limitations **(1)**  *Challenges Expected*  Delimitation **(1)**  *To do to counter anticipated challenges* | **(13 pts)** |  |  |
| Literature Review/Related Work Objectives mapping to Literature Review **(2)**  Critique of Theoretical framework and content adequacy (**2**)  *Principles, parameters of consideration*  Discussion of technologies contextualization for the proposed work **(2)**  Citations of content and alignment to work **(2)**  Review of at least 3 systems comprehensively the working behind it **(2)**  Gaps identification, analysis relative to the proposed solution **(1)**  Conceptual Framework clear to communicate how it works, data flows, processing, actors **(3)**  *Diagram that’s clear; discussion of diagram.*  *Describe input process output storage boundaries.* | **(14 pts)** |  |  |
| Methodology Methodology and justification (**2**)  Correct Methodology Application (**1**),  Design and Development tools (**2**)  Deliverables and milestones **(2)**  *Examinable bits from ideation*  *Proposal, design, test cases documentation doc*  *Proof of concept- modules*  Gantt Chart that makes sense relative to the project **(1)** | **(8 pts)** |  |  |
| Proposal Presentation Table of Contents and List of Figures **(2)**  Are relevant references provided and formatted correctly? **(2)**  Is there a clear and proper use of language? **(1)**  Effective report structure (chapters and sections) and layout **(2)** | **(6 pts)** |  |  |
| Total Marks | **45** |  |  |

|  |  |  |
| --- | --- | --- |
| Verdict (Please tick) | Accept | Reject |

Comments (**Reasons for Reject/Accept**)