UNIVERSITY OF MINES AND TECHNOLOGY

TARKWA

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A THESIS REPORT ENTITLED

SECURE MODEL FOR SOFTWARE QUALITY ASSURANCE

BY

BELINDA IVY BOTCHWAY

SUBMITTED IN FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING

TARKWA, GHANA

FEBRUARY 2022

UNIVERSITY OF MINES AND TECHNOLOGY

TARKWA

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A THESIS REPORT ENTITLED

SECURE MODEL FOR SOFTWARE QUALITY ASSURANCE

BY

BELINDA IVY BOTCHWAY

SUBMITTED IN FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING

THESIS SUPERVISORS

……………………………………

PROF BONIFACE KAYODE ALESE

……………………………………

ASSOC PROF SOLOMON NUNOO

TARKWA, GHANA

FEBRUARY 2022

DECLARATION

I declare that this thesis is my own work. It is being submitted for the Degree of Doctor of Philosophy in Computer Science and Engineering in the University of Mines and Technology (UMaT), Tarkwa. It has not been submitted for any degree or examination in any other University.

....
(Signature of candidate)

….th day of February, 2022.

ABSTRACT

The increasing usage of software in all fields of life, including safety-critical departments of organisations has necessitated the need for the development of quality software. Although software quality is of paramount concern to software development, it may be a challenging task to software developers as it depends on ensuring that developed software meet the standards of software quality design. Different quality models have been proposed by researchers to serve as a benchmark for software quality design but most of these models are tailored towards specific projects' needs, hence, the need for the generic quality model suitable for evaluation of all software projects. In this research, eleven (11) main software quality attributes and thirteen (13) sub-attributes were identified and used for the quality assurance model. These were attained from ten well-known standard software quality models to rank the quality attributes. The Analytic Hierarchy Process (AHP) was used to rank the software quality attributes and maintainability was observed to have the highest score while cost had the lowest score. Mathematical models of the quality attributes were used to evaluate software and the scores attained by each attribute was modeled with a voting model and multiplied with the criteria weight from the AHP to get the overall quality score for each evaluated software. Access to the quality assurance model was restricted by the development of an access control model with the use of the Bell-LaPadula and Biba model to regulate the people who use the model. To perform software evaluation, the application must be hosted online, hence, must have a domain name. As a result, twenty-eight web-based applications, grouped under six (6) categories, namely, Educational, E-commerce, Company, Document Creation Software, Video editing, and Form creation web applications were evaluated. Results from the evaluation showed that Document Creation Software had the highest average quality score of 96.21% while Educational web applications had the lowest average score of 84.16%. To evaluate the performance of the software quality assurance model, recent works were used. The performance evaluation showed that our model outperformed their models when evaluated against the attributes they used and when extended to the use of eleven (11) quality attributes. The access control model was also evaluated for accuracy, precision, and recall and was seen to have values of 0.93, 0.96, and 0.91, respectively. The study has established a model for assessing the quality of software factoring in the major attributes of software quality assurance.

DEDICATION

I dedicate this thesis to the Almighty God and my family.

ACKNOWLEDGEMENTS

My profound gratitude goes to the ever-merciful God for his divine love, peace, guidance, favour, wisdom, knowledge and understanding throughout my academic journey.

My immense appreciation to the Departments of Computer Science and Engineering and Electrical and Electronic Engineering of the University of Mines and Technology (UMaT), Tarkwa, Ghana and the Cybersecurity Department of the Federal University of Technology (FUTA), Akure, Nigeria, for the opportunity granted me to pursue this research.

A special thanks goes to my supervisors, Prof B. K. Alese and Assoc Prof Solomon Nunoo. I am grateful for your guidance, craftmanship and staunch commitment to your roles. There were times that the path was hazy, but your constant voices kept guiding me through this research. It is truly a pleasure to be mentored by you. I am motivated by your in-depth knowledge, unfaltering levels of professionalism and creativity throughout the work.

My profound gratitude goes to the current and immediate past Vice Chancellors of UMaT, Prof Richard Amankwah and Prof J. S. Y. Kuma, Pro Vice Chancellor of UMaT, Assoc Prof Anthony Simons, Dean, School of Postgraduate Studies, Prof Grace Ofori Sarpong, Dean of Quality Assurance, Assoc Prof Bernard Kumi-Boateng, Head, Computer Science and Engineering Department, Dr William Akotam Agangiba, Head, Electrical and Electronic Engineering Department, Dr J. C. Attachie, Assoc Prof (Mrs) Cynthia Borkai Boye, and lecturers of the Departments of Computer Science and Engineering and Electrical and Electronic Engineering for their motivation and continuous support towards the completion of this research work.

I acknowledge the inputs of Assoc Prof (Mrs) Favour-Bethy Aderonke Thompson, Assoc Prof Gabriel Iwasokun, Dr Emmanuel Akinwonmi, Dr Gabriel Arome and Mr Ogundipe Iyanuoluwa Boluwatife towards the success of this research. You supported me in ways that I never expected, and I am forever grateful.

I appreciate my parents, siblings and father in the Lord, Mr. Gill-Christ Essandoh, for your prayers, sound advice, love, motivation and support throughout the entire PhD programme. I am forever grateful. God reward you all bountifully.

TABLE OF CONTENTS

Content	Page
DECLARATION	**i**
ABSTRACT	**ii**
DEDICATION	**iii**
ACKNOWLEDGEMENTS	**iv**
TABLE OF CONTENTS	**v**
LIST OF FIGURES	**viii**
LIST OF TABLES	**x**
CHAPTER 1 GENERAL INTRODUCTION	**1**
1.1 Overview of Research	1
1.2 Problem Definition	4
1.3 Objectives of Research	7
1.4 Methods Used	8
1.5 Contribution to Knowledge	8
1.6 Scope of the Research	9
1.7 Significance of the Research	9
1.8 Research Applications	9
1.9 Facilities Used	9
1.10 Research Approach	9
1.11 Organisation of Thesis	10
CHAPTER 2 LITERATURE REVIEW	**12**
2.1 Introduction	12
2.2 Software	13
2.2.1 Application Software	13
2.2.2 Software Development	15
2.3 Software Quality	16
2.3.1 Software Quality Model	17
2.3.2 Software Quality Attributes	29
2.4 Limitations of the Existing Software Quality Models	33

2.5 Multi-Criteria Decision-Making Analysis 33

2.5.1 Analytic Hierarchy Process 34

2.6 Software Security 39

2.6.1 Software Security Goals 40
2.6.2 Open Web Application Security Project 41
2.6.3 Software Security Model 43

2.7 Voting Model 48

2.8 Summary of Relevant Literature 49

2.9 Research Gaps in Related Works 53

2.10 Justification of Methods Used 53

2.10.1 The Proposed Quality Model 53
2.10.2 The Secured Model 54
2.10.3 AHP Technique 54

CHAPTER 3 SYSTEM DESIGN 55

3.1 Introduction 55

3.2 Quality Attributes and Sub-Attributes 55

3.3 Quality Assurance Model 55

3.4 Assessment of Software Quality Attributes 57

3.5 Mathematical Models for the Secured Quality Assurance Model 61

3.5.1 Mathematical Model for the Access Control Security Model 61
3.5.2 Mathematical Model for the Quality Assurance Model 62

3.6 Voting Method 73

3.7 Architecture of the Proposed Model 74

3.8 Flow Diagram of the Proposed System 76

CHAPTER 4 SYSTEM IMPLEMENTATION, RESULTS, AND DISCUSSIONS80

4.1 Introduction 80

4.2 Identification of Software Quality Attributes 80

4.3.1 Quality Attribute Selection Judgement Matrices 82
4.3.2 Alternative Selection Judgement Matrices 89
4.3.3 Quality models and attributes 96

4.4 Software Quality Assurance Model Implementation 97

4.4.1 Security Model for Quality Assurance Software Access Control 100
4.4.2 Software Efficiency Test 106
4.4.3 Software Reliability Test 110
4.4.4 Software Testability 113
4.4.5 Software Usability Test 116
4.4.6 Software Maintainability Test 150

4.4.7 Software Portability Test 153
4.4.8 Software Functionality Test 157
4.4.9 Software Availability Test 159
4.4.10 Software Reusability Test 161
4.4.11 Software Security Test 164
4.4.12 Software Cost Estimate Test 166

4.5 Voting Method 168

4.6 Performance Evaluation 174

CHAPTER 5 CONCLUSIONS AND RECOMMENDATIONS 178

5.1 Conclusions 178

5.2 Recommendations 179

REFERENCES 180

APPENDICES 200

APPENDIX A CODES USED 200

INDEX 205

LIST OF FIGURES

Figure	Title	Page
2.1	Software Development Life Cycle	16
2.2	Software quality perspectives and software quality factors	18
2.3	Core Security Goals	40
2.4	Bell-LaPadula Model	45
2.5	Artificial Neuron	49
3.1	Hierarchical Structure of the Proposed Quality Assurance Model	56
3.2	Hierarchical Structure of Software Quality Attributes	57
3.3	Voting Method Technique	74
3.4	Architecture of the Proposed Model	75
3.5	Flow Diagram of the Proposed Model	77
3.6	Flow Diagram of the System Login	78
3.7	Flow Diagram of the System Authentication	79
4.1	Weights of the Software Quality Attributes.	88
4.2	Quality attributes and their rates of address by Quality models	96
4.3	User Login Page of the Software Quality Assurance Model	98
4.4	Register Account Page of the Software Quality Assurance Model	99
4.5	Successful Register Account Page	99
4.6	Performance Evaluation of Access Control Model	102
4.7	Homepage of Software Quality Assurance Model	103
4.8	Generated Results showing Scores of Attributes for a Domain Name	104
4.9	Results showing Score from Voting Model	104
4.10	Results for Educational Web Applications	169
4.11	Results for Video editing Web Applications	170

4.12 Results for E-commerce Software 170

4.13 Results for Online Form creation Software 171

4.14 Results for Company Web Applications 172

4.15 Results for Document Creation Software 172

4.16 Average Result of Web-Applications 173

LIST OF TABLES

Table	Title	Page
2.1	Software Quality factors and Quality criteria.	19
2.2	Boehm’s Quality Software Model	21
2.3	Dromey’s Quality Software Model characteristics and attributes	22
2.4	ISO Software Quality Model Characteristics and Attributes	24
2.5	Scale of Comparison	36
2.6	Number of Comparisons with the corresponding RI value	37
2.7	Number of Comparisons	37
2.8	Summary of Literature Review	50
3.1	Scale of Comparison	58
3.2	Number of Comparisons	58
3.3	Questionnaire Given to Expert	58
3.4	Number of Comparisons with the corresponding RI value	60
3.5	Confusion Matrix	62
3.6	Survey Questions and Scale Used	66
3.7	Relationship Between Cronbach Alpha’s Score and Reliability	67
3.8	SUS Score and Grade	68
3.9	Vulnerability Level and Score	70
3.10	Functional Point parameter and Weight of Complexity	71
3.11	Questions for the value adjustment factor	72
3.12	Programming Language and Score Points for AVC	73
3.13	Mode and Factors	73
4.1	Quality Attributes of the Existing Quality Models and the Proposed Model	81

4.2	Geometric Mean of the Filled Questionnaire	84
4.3	Normalised Pair-wise Comparison Matrix	85
4.4	Consistency of Pair-wise Comparison Matrix	87
4.5	The Weight of Alternatives for Maintainability	89
4.6	The Weight of Alternatives for Testability	90
4.7	The Weight of Alternatives for Reliability	90
4.8	The Weight of Alternatives for Efficiency	91
4.9	The Weight of Alternatives for Usability	91
4.10	The Weight of Alternatives for Portability	92
4.11	The Weight of Alternatives for Reusability	92
4.12	The Weight of Alternatives for Functionality	93
4.13	The Weight of Alternatives for Availability	93
4.14	The Weight of Alternatives for Cost	94
4.15	The Weight of Alternatives for Security	94
4.16	The Weights for Software Quality Attribute Selection	95
4.17	Confusion Matrix for Access Control	101
4.18	List of Web Applications being evaluated	105
4.19	Web Applications and Overall Efficiency	107
4.20	Reliability Test Evaluation	111
4.21	Web Applications and Testability Test Rank	114
4.22	Case Processing Summary for Survey	117
4.23	Reliability Statistics of Survey Score	118
4.24	Survey Statistics for Application 1	120
4.25	Survey Statistics for Application 2	121
4.26	Survey Statistics for Application 3	122

4.27	Survey Statistics for Application 4	123
4.28	Survey Statistics for Application 5	124
4.29	Survey Statistics for Application 6	125
4.30	Survey Statistics for Application 7	126
4.31	Survey Statistics for Application 8	127
4.32	Survey Statistics for Application 9	128
4.33	Survey Statistics for Application 10	129
4.34	Survey Statistics for Application 11	130
4.35	Survey Statistics for Application 12	131
4.36	Survey Statistics for Application 13	132
4.37	Survey Statistics for Application 14	133
4.38	Survey Statistics for Application 15	134
4.39	Survey Statistics for Application 16	135
4.40	Survey Statistics for Application 17	136
4.41	Survey Statistics for Application 18	137
4.42	Survey Statistics for Application 19	138
4.43	Survey Statistics for Application 20	139
4.44	Survey Statistics for Application 21	140
4.45	Survey Statistics for Application 22	141
4.46	Survey Statistics for Application 23	142
4.47	Survey Statistics for Application 24	143
4.48	Survey Statistics for Application 25	144
4.49	Survey Statistics for Application 26	145
4.50	Survey Statistics for Application 27	146
4.51	Survey Statistics for Application 28	147

4.52 Overall Usability Evaluation 148
4.53 Software Maintainability Evaluation 151
4.54 Software Portability Evaluation 154
4.55 Software Functionality Evaluation 157
4.56 Software Availability Evaluation 159
4.57 Software Reusability Evaluation 162
4.58 Security Test Results 164
4.59 Cost Evaluation based on software category 166
4.61 Comparison of Proposed Model with Bayu and Banowosari (2021) 175
4.62 Comparison of Proposed Model with Kaur, Kaur and Kaur (2016) 176
4.63 Comparison of Proposed Model with Budiman *et al*. (2018) 177

CHAPTER 1

GENERAL INTRODUCTION

1.1 Overview of Research

Organisations worldwide are adopting various techniques for profit maximisation through software usage and, thus, has become prevalent in aiding these techniques (Saini *et al*., 2020). The emergence of software has eased our way of life by allowing people to perform repetitive tasks easily and faster. It has changed the way jobs are coordinated in the working environment and has positively impacted the global economy due to increase in innovations that has equally enhanced productivity (Salleh, Bahari and Zakaria, 2017). Software has found its application in various fields of life including safety-critical departments. Consequent upon the aforementioned, is the need for quality since the use of less quality software may be prone to adverse effects such as loss of life, financial loss, and mission failure (Sahu and Srivastava, 2018). Software quality is therefore, a key element in software engineering since software users perceive software as a supporting tool in all fields and life.

In that regard, it is of essence that software is reliable and useful as established by Kabir, Rehman and Majumdar (2016) that the rejection of less quality software is on the increase. Therefore, in recent times, most software development companies seek to enhance software quality by incorporating quality standards into software development to meet both user and stakeholder requirements (Gambo, Soriyan, and Achimugu, 2011), yet some other software developers only factor quality when developing safety-critical systems.

Quality is, therefore, the totality of standard features and characteristics of a product to satisfy given requirements (Kabir, Rehman and Majumdar, 2016). According to Tomov and Ivanova (2015), quality can be explained from five (5) different perspectives: transcendental, product, user, manufacturing, and value-based views. The following provide a basic explanation of each of these perspectives:

a) Transcendental view: Quality is seen as a feature that can be predicted yet cannot be explained;
b) User view: Quality is perceived to be the appropriateness for usage;
c) Manufacturing view: Quality is the conformance to requirements;
d) Product view: Quality is perceived to be the essential features of a product; and

e) Value-based view: Quality is seen to have a direct reflection on the amount a customer is willing to pay for a product.

Irrespective of the way quality is defined, it is vital in software development. Software quality according to Hussain, Farid and Mumtaz (2019), refers to the conformance to a specification and meeting customer requirements. Meeting customer requirements is independent of quantifiable attributes. Software quality is, therefore, a standard for measuring the requirements of software to satisfy user's prospects. Software users expect software product to be of high-quality standard and opine that software companies will follow the standards of developing quality software to satisfy sought needs (Kassie and Singh, 2020).

According to Mishra and Otaiwi (2020), the ability of software developers in recent times to create new dynamic and innovative software features of high quality is a critical factor in the software development industry. Developers rely on the adoption of libraries and components of existing software. This exhibits some major drawbacks, which according to Morgenstern, Marx and Landesman (2005) and Al-Badareen *et al*. (2011) are:

a) Vulnerabilities in developed software product;
b) Likelihood of information theft and modification;
c) The developed software becomes unreliable;
d) Customer satisfaction is not met;
e) The cost of maintenance is high;
f) Likelihood of unauthorised access; and
g) Non-compliance to laid down standards.

Research gaps have, therefore, been found in the field of software quality (Pohl and Hof, 2015). Software quality can be evaluated by using software quality models and these models factor quality attributes for determining the quality of software. There are five (5) commonly known software quality models, according to Kassie and Singh (2020), which are: FURPS quality model, McCall's quality model, ISO 9126 quality model, Dromey's quality model, and Boehm's quality model. However, most of these models, if not all, have remained theoretical as there are no visible implementation of designs of any of them (Kaur, 2012).

The ISO 9126 standard states that a quality model must have one or more of the following attributes:

a) Functionality: The presence of functions and their itemised properties. The functions meet specified requirements;
b) Reliability: Ability of a software product to preserve its performance level given a specified condition and time;
c) Usability: The effort needed for use, and on the individual assessment of such use by a stated or implied set of users;
d) Efficiency: The connection between software performance and used resources, under stated conditions;
e) Maintainability: The effort required to make specified improved adjustments; and
f) Portability: The ability of software to be transported from one computing environment to another.

Software security is one of the software quality attributes and refers to the conservation of confidentiality, integrity, and availability of information (Suveetha and Manju, 2016). It is an idea incorporated into software development for safety against malicious attacks to ensure the correct functioning of the software product. In the past, programmers presumed that securing an organisation's infrastructure could prevent malicious attacks but recently, hackers are bypassing software security with techniques such as cross-site scripting and Structured Query Language (SQL) injection attacks (Singh *et al.*, 2015). Hackers or intruders capitalise on system vulnerabilities to render software incapacitated, thus, adopting strict security mechanisms into software development should be keenly ensured in all software usage environments.

Software usability is a significant software quality attribute as well as being a vital attribute for software development. It is used to measure software accessibility; therefore, usability of software must be assessed aptly and regularly to satisfy user needs (Qui, Chui and Helander, 2006). According to Nielsen (2012), usability refers to the ability to easily use and understand a software product. Software is developed for organisations to guarantee profitability, suitability, and accessibility (Signore, 2005); therefore, usability evaluation is significant to improve performance and speed (Islam and Tsuji, 2011).

Software functionality refers to satisfying software stated needs (Dubey, Ghosh and Rana, 2012) and is one of the key quality attributes. It reflects the degree of design compliance.

However, most development processes have not given much consideration to it (Salleh, Bahari and Zakaria, 2017).

Software testability is the verification of the correctness of software to reveal the tendency of flaws in its codes (Nasrabadi and Parsa, 2021). According to Kasisopha, Rongviriyapanish and Meananeatra (2020), there are numerous software measurement methods available for software testing, however, the best method that will suit the project needs to be considered.

The proposed secure model for software quality assurance attempts to design a secure quality assurance software model that meets certain required specifications for software development.

1.2 Problem Definition

In the quest to carry out daily activities with ease, researchers have adopted strategies that have been found over the years to effectively assist mankind. Out of these strategies evolved the development of software, which are used by individuals and organisations. Software may be used to store sensitive information, track performance records in human resources, monitor financial processes, and control workflow, among others in the corporate environment.

The use of software has been trusted to the extent that users do not consider its quality features but solely look out for the efficiency and ability to solve the problem at hand. According to Christakis and Bird (2016), software engineering practices emphasise functionality over quality. Presently, the ability of software developers to frequently offer novel software functionality and features of high quality is debatable in the software development industry (Mishra and Otaiwi, 2020). This is as a result of the increase in demand for software with similar features and functionality, hence, the birth of the concept of software reuse.

Software Reuse is the integration and use of various software components, software libraries, and modules from previously developed software for the development of new applications. Most software development companies rely on reusable software components to decrease development time and cost to improve productivity (Ali, Daneth and Hong, 2020). Despite the known advantages of software reuse, there exist some drawbacks as quality cannot be guaranteed. Additionally, it has the potential of introducing flaws into newly developed applications, hence, the concern on quality arising from the current ways of developing software applications. Most developers rely on software reuse to quickly meet customer demands, thereby, getting good customer feedback when the software is released into the market (Panagiotou and Mentzas, 2011).

In that regard, some developed software applications are not given much consideration to the quality mechanisms put in place. The use of less quality software has a major threat to individuals, companies, and the nation at large as it can directly endanger user's life. According to Petersen (2021), Boeing 777 aircraft had been involved in 31 cases of aviation accidents, including 7 hull losses with 541 fatalities as of February 2021, due to a flawed software algorithm. Also, Boeing 737 aircraft was reported by Campbell (2019) to have experienced a malfunction in its software, which led to another aviation incident.

Another software accident occurred, as stated by Fleischman and Crawford (2020), involving Therac-25, a radiation therapy machine used for the treatment of cancer. Therac-25 was charged with administering an overdose of massive radiation to multiple patients, resulting in their deaths (Johnston, 2021). From all these occurrences, it is evident that software developers and users are much focused on performance and functionalities rather than the quality of the system.

According to Weiss *et al*. (2021), a software error, which was as a result of less quality software, caused the failure of an American air defence system in detecting and seizing an Iraqi scud missile, which resulted in the death of twenty-eight (28) US soldiers with ninety-eight (98) wounded. A software error was also reported by Johnston (2021) to have caused an autonomous Uber vehicle to kill a woman in March 2019. Furthermore, the loss of NASA's Mars Climate Orbiter was associated with failure to use metric units in its software file, which resulted in errors in the system (Buckleton *et al*., 2020). Although the impact of using less quality software is high, some developed software are not giving much consideration to the quality measures put in place. Therefore, there is a need for the development of a quality assurance model that can be used by software developers and designers to evaluate the quality of software.

Despite the increasing urge to provide quality software by adopting various forms of techniques, there are existing models such as ISO 9126, McCall's, FURPS, Boehm's, among others, for evaluating the quality of software.

Related works were reviewed to identify some thought-provoking hitches in the existing software quality models. The limitations were also addressed, and the salient points of these works were documented.

Kassie and Singh (2020) studied software quality factors to enhance software quality assurance. The research was conducted by studying existing software quality models and gathering twenty-seven (27) quality factors. They further conducted a survey by giving out a questionnaire containing twenty-seven (27) questions to seventy (70) participants. Each question in the survey represented one software quality attribute. The participants were to complete the survey for three (3) different software programs, namely, Matrix Laboratory (MATLAB), Microsoft Word, and Mozilla Firefox with different levels of users. They gathered the responses and identified the ten (10) most important software quality attributes that are of importance to users as functionality, operability, usability, portability, reliability, maintainability, understandability, interoperability, efficiency and aesthetic. The limitation of the model was its inability to address other important users' perspective-based software quality attributes such as availability and testability.

In order to prevent an occurrence of compromising the developed software quality assurance model, the overall system is secured by an access control model, which was designed using the Bell-LaPadula (BLP) model for confidentiality enforcement and the Biba model for integrity enhancement. Related works on BLP and Biba models were reviewed to identify some thought-provoking hitches as well.

Saravanan and Umamakeswari (2020) also applied the BLP model to protect user data in a cloud environment. An access control matrix was constructed for patient's records in a hospital using the BLP model where subjects in a particular level, l_i, had access rights to objects, O_j, in the same level. The patient's documents were assigned security values when storing them on the secure cloud storage. Upon retrieval, the credentials were checked and authenticated. Once the authentication process was passed, the user was given access right to the document. Although the user authentication level was successfully ensured, the integrity of user credentials was not enforced.

Liu *et al*. (2017) worked on flexibility enhanced Biba integrity model using break the glass (BTG) strategy for securing operating systems. Although the traditional Biba model can keep the integrity of information, however, it has the tendency of blocking the access requests of some subjects leading to a decrease in the system's accessibility. However, enhanced Biba model using BTG strategy allowed the user immediate access to the system when necessary. BTG is based upon a pre-staged emergency user accounts and allows emergency access to the system. The limitation of the study was that, BTG mode is not open to all the subjects in the system.

Although there have been numerous works done in the area of software quality and security models, from literature done, it was noted that this research differs from other works because it focuses on the design of a secured quality assurance software model using a hybrid software quality model from ten (10) standard and well-known quality models.

Twenty-four (24) software quality attributes were sampled and based on deductions made from literature, were reduced to eleven (11) main attributes with thirteen (13) sub-attributes. These main attributes were ranked using Analytic Hierarchy Process (AHP) to obtain their criteria weights. Two (2) software security models were also used for ensuring access control of the quality assurance software.

The outcome of this research has resulted in the successful development of a secure software quality assurance model to evaluate web-based software programs against eleven (11) software quality attributes. Web-based software used by companies, researchers and educational institutions, business environments, and individuals could be tested for quality. This will build up confidence and trust between software developers and end-users about effectiveness and safety of software products.

1.3 Objectives of Research

The specific objectives of the research are to:

a) identify and analyse the various software quality assurance attributes;

b) carry out a multi-criteria decision-making analysis of the software quality attributes based on the output of (a);

c) develop mathematical models for attributes identified based on (a);

d) develop an integrated mathematical model for the quality assurance software based on a multi-criteria decision-making analysis of the software attributes; and

e) evaluate the performance of the integrated mathematical model using some standard metrics.

1.4 Methods Used

The methods used for the achievement of the stated objectives include:

a) literature review on Analytic Hierarchy Process (AHP), software quality, and software security models (BLP and Biba models);

b) identification of the various attributes of quality software, and analysing them to bring together similar ones;

c) carry out a multi-criteria decision-making analysis of the software quality attributes using the Analytic Hierarchy Process;

d) design of mathematical models from the various models aimed at using (b);

e) use the output of AHP to develop a hybrid model; and

f) evaluate the design using some standard metrics.

1.5 Contribution to Knowledge

The study has contributed to knowledge by:

a) establishing a model for assessing the quality of application software factoring in the major attributes of software quality assurance. This will assist software developers and end-users greatly in developing and assessing software quality;

b) ranking software quality attributes to determine attributes that are of great importance to stakeholders during software development.

c) providing a voting model to bring in all the eleven (11) mathematical models of the proposed software quality assurance model.

1.6 Scope of the Research

The research is limited to the:

a) use of mathematical formulation behind the design of quality and secured software;
b) use of web-based applications for the evaluation process; and
c) assessment of the quality of application software.

1.7 Significance of the Research

This research has established a secure model for software quality assurance based on dynamic analysis that can detect the quality of web-based software.

1.8 Research Applications

The research may be applied in Software Re-use techniques, Software Re-engineering techniques, the design of Commercial-Off-The-Shelf (COTs) software, Enterprise Resource Planning (ERP) software, Service-Oriented software, web-based software applications, and others.

1.9 Facilities Used

The facilities employed for this project include:

a) Library, internet and computing facilities of University of Mines and Technology, Tarkwa and Federal University of Technology, Akure; and
b) Laptop computer equipped with Python programming software. In addition, the following web technologies were used; ExpressJS, Angular, and NodeJS with MongoDB for data storage.

1.10 Research Approach

Methodology for the achievement of the stated objectives are in four (4) phases. Phase one reviewed related works on software, software quality, analytic hierarchy process, and software security models. Relevant literature on the strengths and limitations of the various models are also highlighted. Expertise in information security, software engineering, software development, mathematics, coupled with experts in other fields were consulted.

Phase two employed the use of the Analytic Hierarchy Process (AHP) to rank the software quality attribute for the design of the quality model. The ranking was made from eleven (11) software quality attributes and three (3) alternatives. This information was used to develop a hierarchical structure with the goal at the top level, the attributes at the second level, and the alternatives at the third level, where a pair-wise comparison was done between the attributes to determine their relative importance to the goal.

Phase three of the research deals with the design of the hybrid software quality model and the access control model. Eleven (11) software quality attributes from various software quality models were employed for the design of the software quality model to address the important software quality factors needed for the evaluation of software.

The model was implemented using Python programming language and the web technologies used were ExpressJS, Angular, and NodeJS with MongoDB as its data storage. The application can run on browsers such as Mozilla Firefox, Google Chrome, Microsoft Edge, Internet Explorer, Safari, and Opera Mini. The Operating System requirements are Windows 7, Windows 8 or Windows 10, and Mac OSX 10.8, 10.9, 10.10, or 10.11. The hardware requirements are a laptop with a processor speed of 2.30 Gigahertz (GHz) or above, a minimum of 4 GB RAM, monitor resolution of 1024x768 or higher, Ethernet connection (LAN) or wireless adapter (WiFi) with a speed of 4 Mbps or higher.

1.11 Organisation of Thesis

The thesis is organised as into five chapters.

Chapter 1 is the introductory chapter. It contains the background to the study, problem definition, objectives of the study, methods used to achieve the objectives, contribution to knowledge, scope of work, significance of the study and the facilities that were available for developing and writing the thesis. This chapter also contains a brief summary of the research methodology and gives the organization of the study.

Chapter 2 reviews related literature on models for quality software development and also discusses the quality attributes of these models. The chapter also discusses a technique for multicriteria decision making analysis known as the Analytic Hierarchy Process and its applications by other researchers relevant to the study. Software Security, its goals and Security Models are also discussed. The summary of relevant literature is also contained in the chapter.

Chapter 3 proposes the software quality assurance model that consists of eleven (11) main attributes and thirteen (13) sub-attributes. It also shows the mathematical models used for the development of both the quality assurance model and the security assess control model. AHP technique is used to perform a multi-criteria decision-making assessment to select a suitable software quality attribute for the development of the quality model. Finally, the voting model is applied in the chapter to multiply values from the score of attributes from the software quality assurance model with the scores attained from the AHP technique.

Chapter 4 covers implementation of the proposed model, the results obtained and the comparative analysis drawn from the research and existing works.

Chapter 5 gives the conclusions and recommendations and future research work that can be undertaken.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In the era of globalisation and connectivity, the urge for an easier and faster way to go about daily activities has become of keen importance to all. The usage of smartphones is on the increase and has led to the development of various software applications. As a result, people rely on software programs to enable them to work with ease. Software and information systems have become necessary in most fields of life, including the health sector, business, military, and also in social networking fields (Abrahamyan *et al*., 2016). It has changed the way jobs are coordinated in the working environment and has positively impacted the worldwide economy (Salleh, Bahari and Zakaria, 2017). Although the use of software has eased the way of life, its quality issues are of major concern to both end-users and software development companies who want users to patronise their products.

Software is important in providing a competitive edge for organisations. There is, therefore, the need for development of quality applications to increase the trust of customers and organizations to share information and perform transactions (Mohammed *et al*., 2016). Although previous works have focused on the functionality and usability features of software and do not consider the quality measures put in place (Christakis and Bird, 2016), the quality of software products is currently considered as an essential feature in software development. Furthermore, systems such as safety-critical, real-time and control systems are sensitive, hence, disregarding quality features during development may lead to adverse effects (Al-Qutaish, 2010).

According to Kassie and Singh (2020), software quality has become an important requirement when it comes to software development and as a result, its assessment and improvement are being highlighted by software development companies. In as much as these companies seek to enhance software quality, some other software developers only factor in quality when developing sensitive systems irrespective of user expectations (Gambo, Soriyan, and Achimugu, 2011). The quality of software may be evaluated through various means including software tests (Budiman *et al*., 2018) and has resulted in the proposal of standard quality models by researchers for assessing the quality properties of software (Galli, Chiclana and Siewe, 2020). Software developers who do not assess the quality of their developed software programs may produce software that does not conform to the approved standards for the development of quality software. Hence, there is the need to develop the urge to create a quality assurance model to test for the quality of software programs.

2.2 Software

Software connotes the collection of instructions that tell the computer how to perform certain tasks and enables the user to interact with computing devices (Anon., 2021a). Software may be divided into programming software, system software and application software.

System software is made up of files and programs that form the operating system (Anon., 2021b). It includes device drivers, compilers, and other utilities that help the computing device run efficiently. System software helps to interface the computer hardware with the application software and runs at the lowest level of computers (Anon., 2021b). Programming software is a subset of system software and is a set of tools (compilers, interpreters, and debuggers) or software that helps developers to design application software (Thomas, 2020). Application software is designed to enable users to perform tasks such as creating documents, playing games, and surfing the web. Application software is task-specific and can be simple or complex (Pedamkar, 2020). It resides above the system software and includes programs such as photo editor, word processor, media player, and others.

2.2.1 Application Software

Application software can be either desktop-based, web-based, or mobile-based. According to Bychkov (2013), desktop-based applications usually run locally on a computer system while web-based applications run on two computers, i.e., the server, which is always connected to the internet to provide a unique address called Uniform Resource Locator (URL), and the client computer, which is randomly connected to the internet. Mobile-based applications are designed to operate on mobile devices such as smartphone, tablet or watch rather than desktop or laptop computers.

Desktop-based Application Software

Software development, according to Smith (2021a), began with desktop-based applications which ran directly on the operating system and can be used on standalone machines only. These applications have to be installed separately on multiple client computers and store most of the user-generated data and the application's data on the hard drive of the computer it is running on. There are usability constraints with desktop application usage since it cannot be accessed everywhere, yet, they have an advantage with connectivity since they are standalone and do not face hindrances from internet connectivity (Smith, 2021a).

Mobile-based Application Software

Mobile applications provide users with similar services provided by desktop and web applications. These applications provide limited and isolated functionality such as games or calculators. The simplest mobile application is taken from a desktop-based application and ported into a mobile device while a more sophisticated one is specifically developed for the mobile environment (Anon., 2021c). Mobile applications use iOS or Android as their platform.

Web-based Application Software

According to O'Shea (2017), a web-based application runs over a network such as an internet or intranet. It has features and functionality that are not different from that of mobile-based applications and uses a different platform (iOS or Android is used for mobile applications and web browser is used for web-based applications). They may be programmed using a programming language like JavaScript which has support for web browsers together with a markup language like HyperText Markup Language (HTML). Web-based applications have the capability of updating and maintaining web applications without necessarily distributing and installing software on numerous computers (as in the case of desktop-based applications). It has gained popularity due to the support for cross-platform compatibility. Traditional desktop-based applications are being replaced by web-based applications for portability and easy accessibility since web applications store most of their data on the cloud (Anon., 2021d).

Therefore, this research evaluates the quality of **application software**, which is being hosted online by inputting their Uniform Resource Locators (URLs) into the proposed software quality assurance model for assessment. The research also performed the quality evaluation on the **homepages** of the web applications since errors on homepages mostly run through the rest of a web application (Kurt, 2011).

2.2.2 Software Development

Software development is an iterative process used to create software in an orderly way to address a specific goal. It is carried out by a software programmer through the writing of program codes. Software development goes through various stages such as conducting research to identify the required software needed to perform a task, specifying needed requirements, drawing a software design to include a flow diagram that will encompass the flow of data and processes, and documenting the processes used (Salve, Samreen and Khatri-Valmik, 2018). This is known as the Software Development Life Cycle (SDLC).

Software Development Life Cycle

SDLC is the framework that defines tasks performed at each step in a software development process. According to Mohino *et al.* (2019), the software development process is divided into distinct phases, as shown in Figure 2.1, to improve the software design and ensure that good software is built. These phases, according to Salve, Samreen and Khatri-Valmik (2018), include:

a) Requirement specification: This is mostly the fundamental stage in software development, where customer requirements are gathered. Afterward, the goals, objectives, and estimated cost of the project are documented;
b) System design: This stage defines model formulation, the project architecture, flow of data, flow of processes, and interfacing of components to meet customer requirements;
c) Coding: This is where the developer writes program codes according to how the client wants it to function;
d) Testing: Testing is performed during all the stages in development;
e) Documentation: Documentation of the development process is needed for future reference and becomes handy when changes have to be made in system requirements;

f) Deployment: This comprises activities that make software readily available for use in a given environment. These activities include installation, configuration, and updating; and

g) Maintenance: Maintenance is done to improve software and add new user requirements.

SDLC defines the methods used to improve the quality of software as well as the overall software development (Omar and Fahad, 2017).

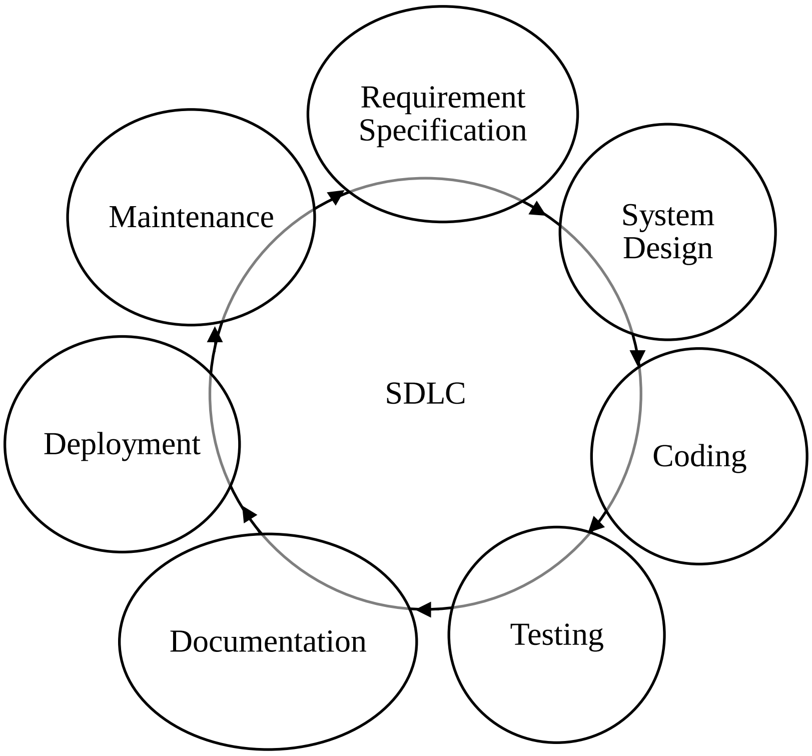


Figure 2.1 Software Development Life Cycle (Mohino *et al*., 2019)

2.3 Software Quality

Software quality is of major concern to software stakeholders as customer demand is equally on the rise (Al-Qutaish, 2010). It measures the degree to which software is designed and its conformance to the design specifications (Hussain, Farid and Mumtaz, 2019). To evaluate that developed software meet user’s stipulated requirements, software quality is the benchmark used. It ensures that user requirements are met, documentation is provided, system design is made and all the requirements that are necessary in developing standard and satisfactory software is followed. It strictly follows the software development life cycle to evaluate and improve software performance (Omar and Fahad, 2017).

There exist various definitions for software quality by software quality experts like Crosby (1979), Juran (1988), Ishikawa (1989), Shewhart (1931), Feigenbaum (1991), and others. According to Crosby (1979), quality means goodness, luxury, or shininess and is used to signify the worth of something. Juran (1988) defines quality as the product features which meet customer needs, thereby providing product satisfaction. Shewhart (1931) defines quality in two aspects, i.e., an objective reality independent of the existence of man, and a result of the objective reality, which relates to what we think, feel or sense. In other words, there is a subjective side of quality.

2.3.1 Software Quality Model

Software quality model is a model that ensures that developed software programs conform to the standard quality of software, and evaluates software using quality attributes. These quality models and attributes play an essential role in the assessment of software quality. Over the years, different quality models have been presented by researchers such as McCall *et al*. (1977), Boehm *et al*. (1978), Deming (1986), Glib (1988), Grady (1992), Dromey (1995), Anon. (2001), and Jamwal and Jamwal (2009). These models have varying attributes, and sub-attributes for evaluating the quality of software. Quality attributes are used to characterise software products and are evaluated using quantitative or qualitative approaches.

McCall Software Quality Model

McCall *et al*. (1977) presented a quality model for measuring software quality known as McCall's software quality model. It is one of the predecessors of today's quality models and is also known as General Electric's Model of 1977 (Tripathi, 2014). It was developed to evaluate the quality of the United States military Air Force system development process and was used by the system developers (Al-Obaithani and Ameen, 2018). In this quality model, McCall attempted to bridge the gap between users and developers by focusing on several software quality attributes that reflect both the users' views and the developers' priorities (Al-Obaithani and Ameen, 2018).

The motivation behind McCall's model was to assess the relationship between external quality factors which are measured by customers; and product quality criteria which are also measured by the software developers (Lisa, 2001). According to Al-Badareen *et al.* (2011), the McCall software quality model defines software quality based on three (3) perspectives: product revision, product transition, and product operation.

a) Product revision: This connotes the ability of making changes;
b) Product transition: This is the ability to of a software product to adapt to new environments; and
c) Product operations: This represents the ease of use, operation and understanding.

The three (3) software quality perspectives are matched unto eleven (11) quality attributes (Fawareh, 2020) as shown in Figure 2.2. These software quality attributes are portability, testability, maintainability, flexibility, integrity, interoperability, efficiency, reliability, usability, reusability, and correctness and are matched unto twenty-three sub-attributes as shown in Table 2.1.

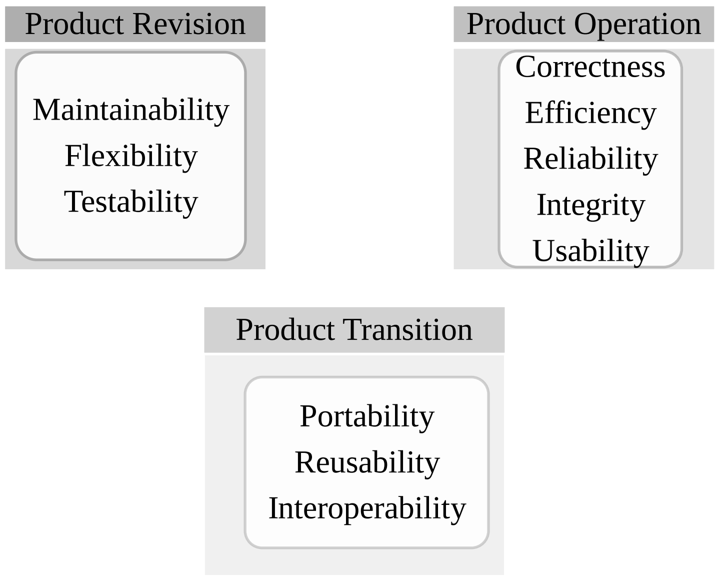


Figure 2.2 Software quality perspectives and software quality factors

Table 2.1 Software Quality factors and Quality criteria.

Quality Attributes/ Factors	Sub-Attributes/ Criteria
Maintainability	Modularity
	Simplicity
	Self-descriptiveness
	Conciseness
Flexibility	Expandability
	Generality
	Self-descriptiveness
Testability	Instrumentation
	Simplicity
	Modularity
	Self-descriptiveness
Correctness	Completeness
	Consistency
	Traceability
Efficiency	Execution Efficiency
	Storage Efficiency
Reliability	Error Tolerance
	Consistency
	Accuracy
Integrity	Access Audit
	Access Control
Usability	Training
	Communicativeness
	Operability
Portability	Machine Independence
	Self-descriptiveness
	Software System Independence
Reusability	Modularity
	Software System Independence
	Generality
	Self-descriptiveness
	Machine Independence
Interoperability	Data Commonality
	Communication Commonality
	Modularity

(Source: Al-Qutaish, 2010)

Although McCall's model creates a affiliation between software quality attributes and sub-attributes, it does not consider the software's functionality (Regan *et al.*, 2020; Waliaro, Omieno and Ondulo, 2019).

Boehm Software Quality Model

According to Olav (2018), Boehm's model is an ordered model that is structured with primitive level characteristics, intermediate level characteristics, and high-level characteristics, which collectively result in the formation of a quality model. The high-level characteristic deals with the general utility, maintainability, and portability of a system (Kassie and Singh, 2020). The intermediate level deals with the flexibility, reliability, efficiency, testability, understandability, and usability of a system. The primitive level provides the foundation for defining quality metrics. According to Al-Badareen *et al.* (2011), Boehm's model added some distinct characteristics to McCall's model but focuses mainly on addressing software maintainability and evaluating the software for its utility. The high-level characteristics deal with the questions mostly asked by software users, such as:

a) General utility/ as-is utility: How well can the software be used as it is?
b) Maintainability; How well can the software be maintained? and
c) Portability: Can the software operate on other computing environments?

The intermediate level deals with the seven (7) quality attributes that represent the qualities a software is expected to have. These include flexibility, efficiency, portability, reliability, testability, understandability, and usability.

The primitive characteristics serve as a guide for defining sub-attributes. This model is created from a wider scope of attributes and integrates nineteen (19) sub-attributes to form the primitive characteristics. The use of primitive characteristics was one of the vital goals established by Boehm when he proposed the quality model. The quality model is shown in Table 2.2.

Table 2.2 Boehm's Quality Software Model

	High-Level Characteristics	**Intermediate Level Characteristics**	**Primitive Characteristics**
General Utility	Portability	Portability	Self-Containedness
			Device Independence
	As-Is Utility	Reliability	Accuracy
			Self-Containedness
			Completeness
			Consistency
			Robustness/Integrity
		Efficiency	Device Efficiency
			Accessibility
			Accountability
		Human Engineering	Communicativenes
			Robustness/Integrity
			Accessibility
	Maintainability	Testability	Communicativeness
			Structuredness
			Self-Descriptiveness
			Accountability
			Accessibility
		Understandability	Conciseness
			Legibility
			Self-Descriptiveness
			Consistency
		Modifiability	Augmentability
			Structuredness

(Source: Al-Obaithani and Ameen, 2018)

Although Boehm's model includes attributes of hardware performance (Nihal and Abran, 2001), it has a major drawback where it did not provide suggestions about measuring these quality attributes and did not cover software functionality (Waliaro, Omieno and Ondulo, 2019).

Dromey Software Quality Model

Dromey's software quality model was proposed to evaluate the requirement determination, design, and implementation phases of software (Dromey, 1995). The model consists of eight (8) high-level quality attributes which include reusability, process maturity, and the six (6) quality attributes from ISO 9126. The framework consists of the design quality model, required quality model, and implementation quality model. In this model, characteristics of software product properties include contextual, internal, correctness, and descriptive, as shown in Table 2.3.

Table 2.3 Dromey's Quality Software Model characteristics and attributes

Software Product	Software Product Properties	Quality Attributes
Implementation	Correctness	Functionality
		Reliability
	Internal	Maintainability
		Efficiency
		Reliability
	Contextual	Maintainability
		Reusability
		Portability
		Reliability
	Descriptive	Maintainability
		Reusability
		Portability
		Usability

(Source: Al-Obaithani and Ameen, 2018)

The main objective for creating this model was to increase the relationship between the attributes and sub-attributes of software quality whiles addressing software properties that affect these attributes (Maryoly, Perez and Rojas, 2002). The limitation of Dromey's model is that it does not address the reliability and maintainability of software products (Fahmy *et al*., 2012).

FURPS Software Quality Model

The FURPS software quality model was proposed by Robert Grady and Hewlett Packard in 1987. FURPS model stands for **F**unctionality, **U**sability, **R**eliability, **P**erformance, and **S**upportability. The model categorises software attributes into functional and non-functional requirements. The functional requirements are defined by the input and expected output, while non-functional requirements include performance, reliability, usability, and supportability. The limitation of the FURPS model is that it did not address software portability and integrity problems (Waliaro, Omieno and Ondulo, 2019).

ISO 9126 Software Quality Model

ISO 9126 software quality model is an internationally accepted model for assessing the quality of software using the internal and external software qualities and their connection to attributes (Nistala, Nori and Reddy, 2019). It comprises of four (4) parts, namely, quality model, quality in use metrics, internal metrics, and external metrics.

According to Hussain, Farid and Mumtaz (2019), the first part of the model is an extension of previous works done by FURPS, Boehm (1978), and McCall (1977). It categorises quality attributes into high-level attributes: portability, reliability, functionality, efficiency, usability, and maintainability, which are broken down into sub-attributes as shown in Table 2.4. The objective of the model is to identify the internal and external software quality attributes. The limitation is that it does not show how the internal and the external software attributes can be measured (Maryoly, Perez and Rojas, 2002). Many scholars have adopted this model for evaluating systems. Such systems include e-book systems (Fahmy *et al*., 2012), website electronic learning systems (Padayachee, Kotze and Van-Der-Merwe, 2010), computer-based systems (Valenti, Cucchiarelli and Panti, 2002), and electronic government systems (Quirchmayr, Funilkul and Chutimaskul, 2007).

Table 2.4 ISO Software Quality Model Characteristics and Attributes

Independent High-Level Quality Characteristics	Software Quality Attributes
Functionality	Suitability
	Accuracy
	Security
	Interoperability
	Compliance
Reliability	Maturity
	Fault Tolerance
	Recoverability
Usability	Understandability
	Learnability
	Operability
	Compliance
Efficiency	Time behaviour
	Resource behaviour
Maintainability	Analysability
	Changeability
	Stability
	Testability
Portability	Adaptability
	Installability
	Conformance
	Replaceability

(Source: Al-Obaithani and Ameen, 2018)

Various works by researchers have seen the application of these quality models in the evaluation of software for quality.

Yadav and Kishan (2020) predicted the reliability of component-based software by analysing and assessing existing software quality models. The assessment was done from models such as McCall's, Boehm's, FURPS, ISO 9126, and Dromey's and a performed analysis showed that software reliability was addressed by all the quality models under the scope. The researchers also observed that software can be easily redesigned using the component-based software engineering approach, hence, the quality of reusable components is of essence. To build a software reliability prediction model, the researchers disclosed that components have to be selected based on the specifications of the model. Therefore, they proposed the sub-attributes of reliability to include maturity, recoverability, and fault-tolerance. The study also presented the parameters for calculating each of the outlined reliability sub-attributes but did not perform an evaluation of the model.

Sharma and Dubey (2015) worked on software reliability by performing a study of the various methods used in literature and extending the methodologies used. They were of the view that the effectiveness of any developed software was dependent on its reliable nature, therefore, software reliability evaluation is of essence. The outcome of the analysis showed that software reliability plays a vital role in software quality assessment and object-oriented metrics aid in reliability prediction. The limitation of the study was the failure to outline the object-oriented metrics being referred to.

Parthasarathy *et al.* (2020) used the ISO/IEC 9126 quality model to assess the quality of standard and customised Commercial-Off-The-Shelf (COTS) products. They quantitatively measured the attributes and sub-attributes of the ISO/IEC 9126 quality model. They measured the software quality of the customised COTS product in its pre-customisation stage. Thereafter, they applied the attributes by presenting some projected values. After customisation of the COTS software, they applied the same attributes. This allowed the researchers to analyse the behaviour of the customised product. From the results obtained, it was observed that sub-attributes such as maturity, operability, understandability, changeability, and suitability worked well after the customisation while sub-attributes such as learnability, adaptability, time behaviour, resource behaviour, replaceability, analysability, and conformance were slightly affected. The sub-attributes that were greatly disadvantaged were testability, compliance, fault tolerance, and accuracy. The research was limited to the application of software quality attributes for one module of a customised COTS package.

Thamer, Mohammad and Ahmad (2013) applied the use of the ISO 9126 model in assessing the quality of software in Enterprise Resource Planning (ERP) systems. The implementation of ERP systems allows institutions and organisations to provide quality and productive professional operations. The researchers observed that, higher educational institutions have invested a high percentage of their funds and time in implementing ERP systems. Therefore, the research objective was to propose a model to evaluate the quality of ERP systems in such institutions. The research listed the attributes of existing quality models and compared it with the attributes of ERP systems. The limitation posed by the research was that it did not rank the main quality attributes of the model.

Alanazi *et al.* (2019) also proposed a quality model to evaluate ERP systems based on the ISO 9126 model. The model was proposed to comprise of reliability, functionality, efficiency, maintainability, portability and usability. The limitation of the model was that, it failed to consider some vital software quality attributes such as flexibility, testability, and availability.

Kabir, Rehman and Majumdar (2016) investigated software usability quality factors. They were of the view that in order to improve quality, it is essential to ensure quality attributes such as learnability, efficiency, usability, and many more. The objective of their research was to analyse ten quality models to propose an improved usability model. The researchers proposed the new usability evaluation model from ten (10) models of Shackel, McCall, Nielsen, ISO 9126, Boehm, ISO 9242-11, FURPS, SUMI, and QUIM models with twelve (12) proposed quality factors, namely, effectiveness, operability, training, attractiveness, satisfaction, usability compliance, reliability, efficiency, understandability, helpfulness, learnability, and human engineering. The contribution made to knowledge was the analysis and comparison of ten (10) recognised quality models to propose an improved usability model that provides twelve (12) category-based usability factors. The limitation was that the research did not show the implementation of the model and, hence, the performance of the model was not evaluated.

Kassie and Singh (2020) studied software quality factors to enhance software quality assurance. The research was conducted by studying existing software quality models and gathering twenty-seven (27) quality attributes. They went ahead to conduct a survey by giving out a questionnaire containing twenty-seven (27) questions to seventy (70) participants. Each question in the survey represented one software quality attribute. The participants were to complete the survey for three (3) different software programs, which were MATLAB, Mozilla Firefox and MS Word with diverse user levels. They gathered the responses and identified ten (10) vital quality attributes of great importance to users as portability, operability, functionality, interoperability, efficiency, maintainability, aesthetic, understandability, reliability, and usability. The limitation of the model was its inability to address other important users' perspective-based software quality attributes such as availability and testability.

Al-Nawaiseh, Helmy and Khalil (2020) proposed a quality model for Academic Information Systems (AIS). The main objective of the research was to help academic establishments that seek to use e-learning systems to assess and select the appropriate quality attributes that are vital to the success of the system. The model was proposed to have six (6) main attributes based on the ISO 9126 model. The research was able to build an approach to measure and assess the quality of AIS in universities and several software quality standards to assist programmers, developers, and system analysts in their projects. The limitation of the proposed model was that, it failed to assess the importance of the evaluated quality attributes.

Noe (2017) presented a usability and accessibility evaluation of e-government websites. The researcher employed the use of Google Speed Insight, Pingdom automation tool, Acunetix, and Wave. The outcome of the evaluation showed that the applications were experiencing multiple usability issues such as broken webpage links and longer loading time problems. The recommendations provided based on the outcome of the results were on the how usability and accessibility issues can be improved.

Kous *et al*. (2018) investigated the usability of a library web application using the effectiveness, satisfaction, and efficiency of the application. The researchers applied the used of formal testing approaches such as log analysis, survey approach and the think-aloud protocol. The outcome specified that the respondents gave a low usability score to the evaluated website, signifying that, it cannot be easily navigated. The researchers further presented recommendations for providing a highly usable website. A major drawback was that the study was only performed on the external usability factors of the website.

Sukmasetya, Setiawan and Arumi (2020) evaluated the usability of a university website using a survey-based approach. The questionnaire contained seventeen (17) questions which were administered to ninety-five (95) respondents. The outcome of the survey demonstrated that the website had an easy usage; nonetheless, there existed few drawbacks that required urgent attention. The internal usability factors such as the load time and page size were not considered.

Uska, Wirasasmita and Fahrurrozi (2019) conducted a study to analyse the usability of the New Student Acceptance (NSA) system in SMAN 1 Pringgarata, a senior high school in Indonesia, using methods such as effectiveness, user satisfaction and efficiency. The research adopted the Likert scale and System Usability Scale (SUS) questionnaire. Results from the study disclosed that the system was highly usable and the researchers additionally suggested ways to provide exciting webpages for user satisfaction.

Bayu and Banowosari (2021) carried out a usability analysis on the payroll system of PT Karya Prima Usahatama company. This assessment was done by conducting a survey. The consistency of the questionnaire was calculated by conducting a reliability check using Cronbach Alpha's mathematical method. This was further modelled with SPSS software. The outcome of the study disclosed that the software was readily understood and highly attractive to the respondents. In addition to the usability analysis, a functionality analysis was done using a survey-based approach with three (3) experts in web development. Sixty (60) web functions were assessed and all experts evaluated the functions as "working correctly". The authors also carried out an efficiency analysis using an online automated tool called GTmetrix. Results showed that pagespeed was 84%, Yslow was 65%, fully loaded time was 3.3 seconds, total page size was 336 Kilobytes and page request was 30 seconds. Portability test showed a success rate as the software was seen to easily move between web browsers. Reliability test also showed a success rate when a stress test was conducted within 10 minutes. Software maintainability was evaluated with Land R version instrument and results showed that there was consistency in the form designs and there was a warning on the data processing system to signify errors created by the user. The maintainability test was also seen to have been passed. Although, the software was concluded to be of good functionality, the usability technique used could not assess the software's internal factors. Also, the efficiency test did not evaluate parameters such as the software's throughput and bandwidth.

Rahardjo, Mirchandani and Joshi (2014) performed a functionality evaluation where they assessed the functions and features of e-government websites in Indonesia. They used a survey methodology and identified that functions related to transactions are focused on efficiency and appeal whereas functions for general services focus on appeal, quality, and personalisation. The authors suggested that developing a successful e-government website was dependent on considering the functionality and features that are of importance to the citizens.

Budiman *et al*. (2018) evaluated the quality of a student academic portal based on the ISO software quality model. The authors considered four (4) quality attributes including Usability, Reliability, Efficiency and Portability based on increasing number of users on the portal. In evaluating for software efficiency, the result for page speed was 66%, YS-low grade was 67%, response time was 5.29 seconds and average load time was 5.09 seconds. For portability evaluation, results showed that the portal could be accessed without error and can run on different web browsers without encountering error. Reliability evaluation showed that when the software was simulated for 500 users, there was a reliability score of 100%. For usability analysis, the authors did the evaluation based on heuristic analysis and results showed that the score was in the small problems category. A major drawback to the study was that the usability evaluation method used could not categorically state the score for usability and the software's internal and external usability factors were not assessed.

Kaur, Kaur and Kaur (2016) drew a correlation between the sub-attributes of usability, including, speed, load time, performance, heat maps, user experience, Search Engine Optimisation (SEO), number of requests, page size, security, navigation, content, design, mobile readiness, accessibility, and clickstream of twenty-one (21) online automation tools. The researchers additionally assessed the performance efficiency of various university websites using GTMetrix, Pingdom, Site Speed Checker and Website Grader. Scores attained by the websites were analysed and the overall usability score was shown. The drawback is that the external usability factors of the websites were not evaluated.

2.3.2 Software Quality Attributes

Software quality attributes are the artistic measurements for postulating customer needs of a software and are used by developers. Most of the quality models mention that quality attributes such as performance, efficiency and reliability can be measured by executing the system while other quality attributes such as usability can are observed by system execution. The software development life cycle certifies that the employment of quality attributes in software development may lead to the creation of a well-engineered software, hence, it must be made compulsory in software implementation, development, and deployment phases (Sharma, 2017).

External Software Quality Attributes

External attributes specify the relationship between the environment and the system or process (Martins *et al*., s2020). It deals with how the software product works in the deployed environment and regulates the realisation of stakeholder's specifications. It ensures that the system provides the required functionality with clearness and consistency. It only affects the user of the software. External attributes result from internal attributes.

Correctness: Correctness is the ability of software to meet its stipulated results. McCall defines correctness as the extent to which a program meets its specifications (Tinnaluri, 2016). It determines the degree to which a software's design and implementation are free from defects.

Usability: Usability refers to the ability of customers to easily use, understand and learn software. McCall defines usability as the ease with which a user can navigate through the system (Weichbroth, 2018); Boehm defines it as the reliability, efficiency, and human-engineering of software; Dromey defines it as the capability of the software product to be understood by users; and ISO defines it as a set of attributes that relate to the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users. According to Madan and Dubey (2012), usability is a product attribute that influences the quality of a software system. Nielsen (2012) defines usability with five (5) attributes: learnability, efficiency, memorability, errors, and satisfaction.

Efficiency: Efficiency is the performance of software by accomplishing tasks in at a faster rate, while using fewer resources and saving computer power. McCall defines efficiency as the number of computing sources and code required by a program to perform its function; Boehm defines it as the ability of the software to satisfy its purpose without waste of resources; Dromey defines it as the capability of the software to adequately perform irrespective of the number of resources used; ISO defines it as the degree to which software makes optimum utilisation of the resources (Tinnaluri, 2016).

Reliability: Reliability refers to the likelihood of software to operate in a given environment within a specified period without encountering a breakdown. McCall defines it as the ability of a program to withstand failure; Boehm defines it as the ability of software to perform its intended functionalities satisfactorily (Tripathi, 2014.

Robustness: Robustness refers to the ability of a software product to cope with any form of error it may encounter during operation (Dubey, Ghosh and Rana, 2012).

Functionality: Functionality is the ability of software to perform the tasks for which it was intended. Dromey defines functionality as the capability of the software product to provide functions that meet stated and implied needs when the software is used under specified conditions; FURPS defines it as the totality of feature sets, capabilities, and security of software; and ISO also defines it as the degree to which software satisfies its stated needs (Dubey, Ghosh and Rana, 2012).

Performance: Performance refers to the total effectiveness of a software product. FURPS defines performance as the ability to impose conditions on functional requirements such as speed, efficiency, availability, accuracy, throughput, response time, recovery time, and resource usage.

Availability: Availability refers to the degree to which a software product is operational and easily accessible when needed for usage.

Security: Security is the ability of a software product to reduce the likelihood of malicious attacks and loss of information. It is the measure of a system's ability to resist unauthorized access.

Cost: Software cost is the amount of money paid for software development. It may be charged based on the category of the developed software. Web-based software applications may be grouped under five (5) categories: retail, financial services, news and information, portals and entertainment web applications (Choudhury and Choudhury, 2010). The cost of developing a retail web application according to Smith (2021b), ranges between $20,000 and $210,000; financial web application ranges between $20,000 and $30,000 (Rehman, 2019); news and information web applications range between $2,000 and $9,000 (Anon., 2021e); portals range between $2,500 and $600,000 (Ibanga, 2021); and entertainment web application ranges between $40,000 and above $100,000 (Martin, 2020).

Internal Software Quality Attributes

Internal attributes are derived from the software product. It deals with how the software product was developed and determines a developer's ability to move forward in a project (Nilson, Antinyan and Gren, 2019).

Maintainability: Maintainability is the ease with which software can be modified to correct faults or improve performance. McCall defines maintainability as the effort required to fix and test errors; Boehm defines it as the easiness to modify and test software; Dromey defines it as the capability of the software product to be modified; and ISO defines it as the ease with which the software can be modified.

Flexibility: Flexibility is the ability of software to adapt to possible future changes in its requirements. McCall defines flexibility as the effort required to modify an already operational program; whiles Boehm defines it as the ability of software to facilitate the incorporation of changes once the nature of the desired change has been determined. Highly flexible software applications have modules and components that are loosely coupled.

Portability: Portability is the measure of the ease of transferring software from one computing environment to the other. McCall defines portability as the effort required to port an application from one system to another; Boehm defines it as the ease of software operation on computer configurations other than the one it currently runs on. Dromey defines it as the capability of the software product to be transferred from one environment to another; ISO defines it as the ease with which software can be migrated from one environment to the other.

Reusability: Reusability is the use of existing tested and validated loosely coupled components in the development of software applications. McCall defines it as the extent to which a program or sub-program can be re-used in other applications.

Testability: Testability is the ease with which the correctness of software can be verified. McCall defines testability as the effort required to test a program so that it performs its intended specification in an error-free state. Boehm defines it as the ability to facilitate the establishment of verification criteria and supporting the evaluation of software performance.

Understandability: Understandability is the capability of a software product to enable the user to understand whether the software is suitable and its usability for specific tasks and conditions for use. It has a major influence on cost and reliability when it comes to the maintenance and reuse of the software. Boehm defines it as the clarity of software purpose to the inspector.

Interoperability: Interoperability is the ease with which software is used with other software applications. McCall defines it as the extent required to couple one system to another.

2.4 Limitations of the Existing Software Quality Models

Although researched works on software quality models are good referencing tools for defining product quality, there exist some loopholes that are worth addressing. ISO 9126 model tends to be more precise than the other models, yet, it has not clarified how the quality attributes can be measured (Djouab and Bari, 2016). Also, most of the models tend to ignore certain quality attributes while some also fail to describe how quality attributes can be measured (Kaur, 2012). In the case of McCall's model, software functionality was not considered although it is a very essential attribute (Tabassum *et al*., 2017). As a result, the user's vision is not factored and hence, user requirements are not met. Boehm's model also did not describe how the quality attributes can be measured (Waliaro, Omieno and Ondulo, 2019). Likewise, FURPS model did not consider other equally important software quality attributes such as portability which is vital in software development (Regan *et al*., 2020).

2.5 Multi-Criteria Decision-Making Analysis

Multi-Criteria Decision-Making (MCDM) is a branch of operational research that deals with evaluating multiple conflicting criteria in decision-making (Kumar *et al.*, 2017). It helps to find optimal results in complex situations where there is a need to choose between the alternatives being evaluated. MCDM is applied by individuals, groups, and organisations to perform tasks such as short-listing job applicants, selecting new projects or investments, among others. MCDM has been applied extensively in science and industry to enhance quality decisions by making the process more explicit, rational, and efficient (Lai and Ishizaka, 2019). It is used where there is the need for alternatives to be ranked, prioritised, or chosen based on multiple criteria being considered (Choudhuri, 2014). MCDM reduces the impact of biases from decision-makers relying on their feelings and preferences. It uses weights between criteria in a structured way and, hence, the results obtained from using it are more transparent and consistent (Lai and Ishizaka, 2019). It works by quantifying qualitative criteria and calculating the total score of the evaluation subjects according to the weight of each criterion or alternative (Aliu *et al*., 2020). This helps decision-makers to have a stronger and more accurate basis on the choice to make. MCDM, according to Aliu *et al*. (2020), involves four (4) components:

a) Alternatives: Objects or individuals to be ranked;
b) Criteria: The alternatives to be evaluated and compared;
c) Weights: The relative importance of the criteria; and
d) Decision Makers: The experts whose preferences are to be represented.

Some commonly used MCDM methods, according to Wang *et al*. (2020), include Analytic Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) Hierarchy Process (AHP), Preference Ranking Organisation Method for Enrichment Evaluation (PROMETHEE), and Analytic Network Process (ANP). However, many researchers consider the AHP technique to be well suited for group decision-making (Lai, Wong and Cheung, 2002).

2.5.1 Analytic Hierarchy Process

Analytic Hierarchy Process (AHP) is a technique for multi-criteria assessment that simplifies decision-making processes. It was formerly developed by Thomas L. Saaty (Saaty, 1977) to provide measures of judgement consistency; to derive priorities among criteria and alternatives; and to simplify the rating of preferences among decision criteria using pair-wise comparisons (Khwanruthai, 2012).

AHP is based on mathematics and psychology (Osman *et al*., 2014). It helps decision-makers to find a decision that best suits their goal and their understanding of a given problem. It is a method to derive ratio scales from paired comparisons (Teknomo, 2017) and is based on a certain scale that changes subjective judgements into objective judgement and solves qualitative problems with quantitative analysis. It is simple and hence has seen its application in many fields.

According to Sarkar (2011), the four (4) steps of AHP methodology are:

a) Build a decision hierarchy by breaking down the problem into various components: objective or goal, criteria or attributes, and alternatives;

b) Gather relational data for the decision criteria and encode them using the AHP relational scale;
c) Estimate the relative priorities (weights) of the decision criteria and alternatives; and
d) Perform the composition (synthesis) of priorities of criteria and alternatives, which ranks the alternatives to the problem objective.

The input for AHP decision-making can be attained from real measurements such as price, colour, and others or from subjective opinions such as feelings and preferences. AHP is seen to be one of the best multi-criteria decision-making tools as it allows for small inconsistency in judgement since there may be some levels of inconsistency in human judgement. AHP breaks down complex multi-criteria decision-making problems into hierarchy interrelated decision criteria and decision alternatives. It uses a prioritisation procedure to determine the relative importance of criteria.

In AHP, a problem involving “m” alternatives and “n” attributes is used to form a judgement matrix of alternatives of order m × m and another judgement matrix of order n × n. Further, a decision matrix of order m × n is constructed using the relative scores of the alternatives. The AHP judgement matrix is shown in equation (2.1).

$$A = \begin{bmatrix} 1 & \frac{w_1}{w_2} & \cdots & \frac{w_1}{w_n} \\ \frac{w_2}{w_1} & 1 & \cdots & \frac{w_2}{w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{w_n}{w_1} & \frac{w_n}{w_2} & \cdots & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{2.1}$$

where, A = Comparison pair-wise matrix;

w_1 = weight of element 1;

w_2 = weight of element 2; and

w_n = weight of element n.

The AHP relational scale of real numbers from 1 to 9 and their reciprocals are used to assign preferences in a systematic order. AHP Pair-wise comparison is based on a standardised comparison scale of nine (9) levels as shown in Table 2.5.

Table 2.5 Scale of Comparison

Scale of Importance	Degree of Preference
1	Equal Importance
3	Moderate Importance
5	Strong Importance
7	Very Strong Importance
9	Extreme Importance
2,4,6,8	Intermediate Values
1/3, 1/5, 1/7, 1/9	Values for Inverse Comparison

(Source: Saaty, 2008)

A normalised pair-wise matrix, $X_{i\,j}$, was generated by summing the values, $\alpha_{i\,j}$, in each column of the pair-wise matrix and then dividing each element in the matrix by its column total as shown in equation (2.2).

$$\mathrm{X}_{i\,j} = \frac{a_{i\,j}}{\sum_{i=1}^{n} a_{i\,j}} \tag{2.2}$$

where n = number of columns.

To generate the weighted matrix, $W_{i\,j}$, the sum of the rows of the normalised pair-wise matrix were divided by the number of criteria, *n*, used. This is shown in equation (2.3).

$$W_{i\,j} = \frac{\sum_{j=1}^{n} x_{i\,j}}{n} \tag{2.3}$$

The consistency vector, $\lambda_{\max}$, was calculated by multiplying the pair-wise matrix by the weight vector and the sum of the row entries were divided by the corresponding criterion weight.

To evaluate the consistency of one's judgement, the Consistency Index (CI) is calculated. This is shown in equation (2.4).

$$\mathrm{CI} = \frac{\lambda_{\max} - \mathrm{n}}{\mathrm{n} - 1} \qquad 2.4)$$

where n = order of the matrix.

The Consistency Ratio (CR) is also calculated using equation (2.5).

$$\text{CR} = \frac{CI}{RI} \tag{2.5}$$

where RI = Random Index.

The Random Index (RI) is shown in Table 2.6.

Table 2.6 Number of Comparisons with the corresponding RI value

n	1	2	3	4	5	6	7	8	9	10	11
RI	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.52

If CR$\leq$ 0.1, the judgement is seen to be acceptable, else the judgement is to be re-examined.

In AHP, the number of comparisons is given in Table 2.7.

Table 2.7 Number of Comparisons

Number of Things	1	2	3	4	5	6	7	n
Number of Comparisons	0	1	3	6	10	15	21	$\frac{n(n-1)}{2}$

The AHP process is repeated for the alternatives too. The weighted matrix of the alternatives is multiplied with the weighted matrix of the attributes or criteria.

The structure of the final decision matrix is shown as

$$\begin{array}{cccccc}
 & W_1 & W_2 & W_3 & \cdots & W_n \\
A_1 & a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\
A_2 & a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\
A_3 & a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
A_m & a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn}
\end{array}$$

where, W_1 to W_n = criteria;

A_1 to A_m = alternatives; and

a_{mn} = number in row m and column n.

The overall consistency ratio, $\overline{CR}$, was calculated as shown in equation (2.6) by summing up the weighted consistency index, w_iCI_i, in the nominator and the weighted random consistency index, w_iRI_i, in the denominator.

$$\overline{CR} = \frac{\sum_i w_i CI_i}{\sum_i w_i RI_i} \tag{2.6}$$

AHP has been applied by several researchers to enhance group decisions. It has been applied in the military (Aull-Hyde and Davis, 2012), educational sector (Sharma, Kumar and Grover, 2020), construction projects (Okudan and Budayan, 2020), electronic toll collection systems (Aliu *et al.*, 2020), etc.

Verma and Mehlawat (2017) used the AHP technique to assess and select Commercial-Off-The-Shelf (COTS) components. The authors realised that the technique was valuable in the creation of trade-offs between tangible and intangible factors when evaluating the weight of COTS components. The application of these weights aided in the determination of the best component using some constraints. The major drawback of the approach was that, fewer alternatives were used in the comparison.

Siavvas, Chatzidimitriou and Symeonidis (2017) introduced an adaptive framework called Qatch to assess software product quality. The researchers employed the AHP technique in developing a model that uses statistical analysis to produce a stakeholder-required software quality model. The technique was evaluated to be adequate for decision-making and useful in situations that require the elimination of complexity in a pair-wise comparison matrix. The drawback of the research was the inability to rank the quality attributes used.

AHP was applied by Dubey and Mishra (2014) to evaluate the reliability of object-oriented software based on the ISO/IEC 9126 model. The outcome of their results disclosed AHP to be a valuable tool in making group decisions where there is the necessity to choose between the objects being evaluated.

Febrero, Moraga, and Calero (2017) analysed software reliability based on AHP. The feasibility and rationality of their model was proved by applying it to a large industrial system. This provided a piece of empirical evidence on the conceptual descriptiveness, by capturing stakeholders' views and industrial applicability efficiently. The drawback was the difficulty in computation with an increased number of pair-wise comparisons.

Kumar and Singh (2016) applied AHP to evaluate the aspect-oriented software quality model. The weights for the evaluation process were calculated equivalently between the attributes and sub-attributes using AHP. The researchers concluded that AHP provides a powerful tool that makes decisions in situations involving multiple objectives. The drawback was the ranking irregularities in the scores attained from experts.

Yujun *et al.* (2019) used the AHP technique to perform a risk assessment on software quality. The authors evaluated the weight and order of risk factors to form an index risk assessment of software quality. This led to the classification of risk into technology, process, demand, and management risks. The results showed that process risk is an important source of software quality risk. The drawback was that the survey was performed on a small population, hence, fewer responses were attained.

Aliu *et al*. (2020), used AHP and fuzzy comprehensive evaluation to analyse information security risk. The weights obtained through AHP were used for both the single and multi-level factor analysis of the fuzzy comprehensive evaluation. Results obtained showed that the risk assessment would assist in recommending the necessary controls for information security systems. The survey was performed on a small population; hence, fewer responses were attained.

2.6 Software Security

Security refers the act of guarding computing systems and data being stored or accessed from harm, theft, and unauthorised use. Security can also be defined as the guard against valued resources or computer systems. These resources could be software, hardware, data, infrastructure, processes or people. Security also refers to the amalgamation of access control, confidentiality, authentication, integrity, non-repudiation and availability to either protect an institution, individual or a nation (Alese *et al.*, 2007). Security can be breached through intentional or unintentional means. These intentional and unintentional security violations by end-users also cause severe security losses (Aldabbas and Teufel, 2016).

In 2011, Sony Pictures encountered an SQL Injection attack where about a million of their users' accounts including passwords, emails, and home addresses were released by LulzSec (Poggi, 2018). The worst case was that Sony stored its customer's data in plain text and not in an encrypted format. In 2015, the Vtech Learning Lodge database encountered a security breach that exposed about 6.3 million children's profiles (Shasha *et al*., 2019). In 2015, Uber accidentally revealed the personal information of hundreds of its drivers (Kokalitcheva, 2015). Recently, LinkedIn was also seen to have suffered a security breach in which about 6.5 million user names and passwords were exposed (Johnson, 2016).

For these reasons, security has become an essential component in all phases of software development (Felderer *et al*., 2016), hence, measures have to be put in place to ensure that software programs are properly secured to prevent intrusion and loss or damage of data.

2.6.1 Software Security Goals

The core security goals are confidentiality, integrity, and availability as shown in Figure 2.3.

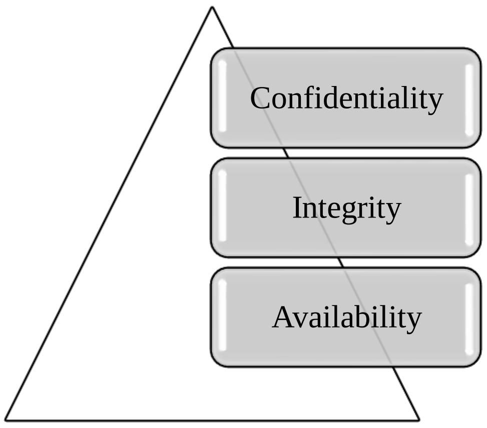


Figure 2.3 Core Security Goals (Source: Dorri *et al*., 2017)

Confidentiality means that people are denied the privilege to access sensitive data either on a computer or data traveling on a network, which they are not entitled to.

Integrity, on the other hand, prevents attackers from destroying and altering data on a computer or data that is traveling on a network.

Availability means that authorised people to data or resources have access to it and can read and modify it.

Software security is a critical issue that needs to be given much attention, hence, the Open Web Application Security Project (OWASP) was founded to evaluate application software security and classify the vulnerability level under ten (10) categorical levels.

2.6.2 Open Web Application Security Project

Open Web Application Security Project (OWASP) is a non-profitable foundation that produces articles, methodologies, documentation, and tools to detect and improve web application security (Ouissem *et al*., 2021). The OWASP ten (10) security vulnerability are classified, according to Pandya and Patel (2016) under Code Injection attack (A_1), Broken Authentication and Session Management (A_2), Cross-Site Scripting attack (XSS) (A_3), Insecure Direct Object References (A_4), Security Misconfiguration (A_5), Sensitive Data Exposure attack (A_6), Missing Function Level Access Control (A_7), Cross-Site Request Forgery attack (CSRF) (A_8), Using Components with Known Vulnerabilities (A_9) and Unvalidated Redirects and Forwards (A_{10}).

Injection (A_1)

Code injection attack, according to Anon. (2021f), occurs due to the sending of unauthorised data to web applications by attackers to make the application perform operations that it has not been originally programmed to do. Injection attack is considered as the most critical web application security risk as it can be easily exploited and can have a severe impact on the data of the hosted application such as data loss or data corruption (Al-Khurafi and Al-Ahmad, 2015).

Broken Authentication and Session Management (A_2)

Broken Authentication and Session Management is a type of vulnerability that allows attackers to steal privileged user accounts to gain control of an application. This vulnerability level is one of the top risks, according to OWASP, and occurs due to the improper implementation of user authentication and active session management (Kelley *et al*., 2012). An example is the 2015 cyber-attack by Pakistan against Bangladesh where about 180 web applications were defaced as a result of broken authentication problems (Hassan *et al*., 2018).

Cross-Site Scripting (XSS) (A_3)

Cross-Site Scripting attack is seen as one of the most predominant web application vulnerability attacks in recent times (Johns, Engelmann and Posegga, 2008). This attack does not affect the server-side but rather occurs within the user’s web browser, hence, affecting the user. It occurs when web pages display inputs that are not properly validated.

As a result, attackers inject malicious scripts into the software to modify the display of content and execute the codes they provide on the computer of any user that visits the web application (Gupta and Gupta, 2017).

Insecure Direct Object References (A_4)

Insecure Direct Object Reference occurs when a programmer refers to an internally implemented object (Ouissem *et al*., 2021). This object may be a database key, directory, or even a file. Hackers can, therefore, bypass the authorisation process when there is no strict security measure to access these unauthorised resources in the database of the system that otherwise would not be accessible (Srinivasan and Sangwan, 2017).

Security Misconfiguration (A_5)

Security Misconfiguration occurs when a component of a system such as a framework, application server, web server, database, and network router is not well designed or configured (Ouissem *et al*., 2021). The use of default configuration of components may lead to vulnerabilities as attackers may exploit these configuration flaws to attack the entire system.

Sensitive Data Exposure (A_6)

A database may be used to store personal data like phone numbers, addresses, login credentials, or credit card details. Sensitive Data Exposure occurs when data that should be protected are compromised (Gorrie, 2021). This may result from weak or no encryption, software errors or unintentionally uploading files to an incorrect database. Once these vulnerabilities are exploited by an attacker, there can be information theft. Information storage web applications that do not use a secure version of the hypertext transfer protocol (HTTPS) as their security are also susceptible to data exposure (Kudkar, 2021).

Missing Function Level Access Control (A_7)

Access rights must be checked before allowing access to some resources in a web application. When users are allowed to perform activities that should be restricted or access resources that should be protected, missing function level access control is said to have occurred (Hamit, 2014). This occurs as a result of flaws in the authorisation logic (Chetan, 2017).

Cross-Site Request Forgery (A_8)

Cross-Site Request Forgery (CSRF) is an attack that makes a user's web browser execute unwanted Hypertext Transfer Protocol (HTTP) requests on a vulnerable web application, thereby, causing an undesired action (Sudhodanan *et al*., 2017). This occurs when attackers use inferred authentication mechanisms of the HTTP protocol and cookies cached in a browser to pass the authentication process and execute the attack on the targeted website (Zhang, Hu and Huo, 2021).

Using Components with Known Vulnerabilities (A_9)

Attackers mostly exploit system vulnerabilities by running automated scripts to probe web applications for known vulnerabilities (Ochaun, 2020). Therefore, the use of components such as APIs, libraries, open-source codes, and frameworks that have previously been successfully exploited in the past endangers a web application (Sundar, 2014).

Unvalidated Redirects and Forwards (A_{10})

Unvalidated redirects and forwards occur when a web application receives input from an anonymous source that could lead to redirecting requests to an untrusted URL (Ayachi *et al*., 2019). Improper validation of web applications could cause an attacker to redirect users to phishing or malware sites.

2.6.3 Software Security Model

A security model refers to the system for implementing security policies. They can also be referred to as methods used to validate security policies to provide procedures that a computer can follow to implement vital security procedures. According to Justiniano (2015), there are several types of security models including the following but not limited to the BLP confidentiality model, Biba integrity model, Harrison- Ruzzo-Ullman model and Graham-Denning model.

Bell- LaPadula Confidentiality model

BLP was the first multilevel security policy model that was designed for military applications. It provides strict protection of confidential information and is mostly used in military environments. BLP policies enforce multi-level security policies to ensure confidentiality requirements and flexibility of access control policies (Ramkumar, 2017; Patel and Sahani, 2018).

BLP model aims to attain multi-level security (MLS) policy by stopping information leakage from subjects in a high-level category to subjects in a low-level category. In order to ensure the multi-level security policy, BLP model outlines two security properties: simple security property (ss-property) and star property (* -property). The ss-property simulates the real world where subjects are denied read privilege to objects with a higher security level. The star property allows subjects the write privilege to objects when their security level is higher than that of the object (Zhu *et al.*, 2016). In the BLP model, information cannot flow towards levels of lower confidentiality because this would cause information leakage (McMillin and Roth, 2017).

BLP model as shown in Figure 2.4 focuses on ensuring that subjects with different clearances are properly authenticated. It uses the simple security rule (no read-up rule) and the star property rule (no write-down rule). It is a state machine model and, hence, defines states with current permissions and current instances of subjects accessing the objects (Cankaya, 2011).

The set of access rights given to a subject are read, append, execute and read-write. Read gives the subject permission to only read the object; Append allows the subject to only "write" to the object but it cannot "read"; Execute allows the subject to execute the object but can neither "read" nor "write"; Read-Write gives the subject both "read" and "write" permissions to the object.

Reading down: A subject at a given security level only has the read access to objects whose security level is below the subject's security level.

Writing up: A subject can append an object whose security level is higher than its security level.

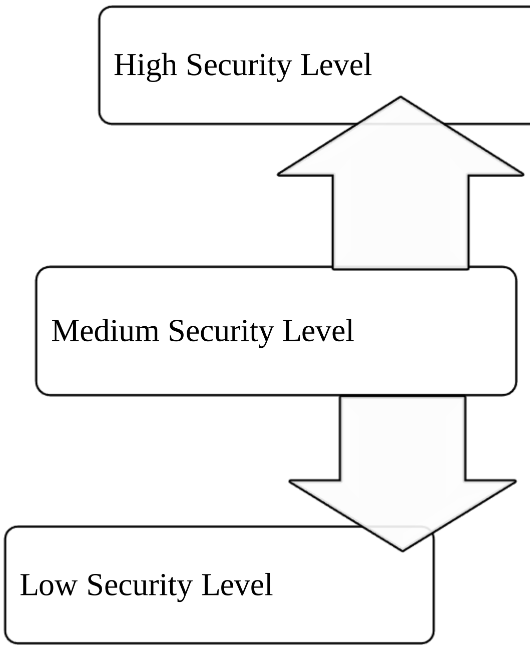


Figure 2.4 Bell-LaPadula Model

Application Areas of BLP Model: The BLP model is used in areas where there is much focus on restricting access control to information. It is used in military applications and government applications due to its rigidity and high cost (Ghosh, Singhal, and Das, 2019). It can also be used to stop virus infection (Zhu *et al*., 2016).

Strengths of BLP Model: The BLP model has the following strengths (Zhu *et al*., 2016):

a) Read Down property: This property prevents users from gaining access to information that is above their security clearance. A user with a low clearance level is not allowed to access information above its clearance level whereas a user with a high clearance level can access information beneath it;
b) Write Up property: Data tends to migrate into higher security classifications, hence, the clearance of a subject attempting access to an object is compared with the object's classification; and
c) Mathematics-based model: It is a mathematical model that uses a set theory to define access rights while keeping a secure operating state.

Problems with BLP Model: The BLP model has the following limitations (Toapanta *et al*., 2018; Liu *et al*., 2016):

a) It addresses only confidentiality giving no regards to integrity or availability;

b) The process of assigning and enforcing security classifications for each user is glossed over in the model and is hard to implement in real life;
c) Information flow is restricted. Lower levels cannot access information of higher classification;
d) Data tends to migrate into higher security classifications; and
e) High level of rigidity.

Balamurugan *et al.* (2015) used the BLP model to secure cloud computing by summarising all the access control techniques in a cloud environment and coming up with a novel attribute-based access control model. The researchers used an enhanced BLP model inspired by the honey bee behaviour. Although they were able to ensure privacy and make users feel secure to store and retrieve data to and from the cloud, data integrity was not enforced in their design.

Salman *et al.* (2017) used the BLP model in a private cloud environment to dynamically change the security level of objects. The researchers presented a multi-level security model with the use of the BLP, which has increasingly seen its application in the information security domain. They reviewed its application in the network domain, and proposed a modified version of the BLP model for the proposed 5G/IoT. The model was proven to be secure by demonstrating the transition from one secure state to another, thereby, conforming to the defined security properties. The drawback of the model was its failure to clearly define the security management tasks.

Biba Integrity Model

Unlike the BLP model, which addresses confidentiality, this is an information flow model that addresses the integrity of data (Henk and Sushil, 2014). The Biba integrity model was published in 1977 at the Mitre Corporation, one year after the BLP model was published. Biba integrity uses a simple integrity rule (no read down), star integrity rule (no write up), and invocation property (Hopkins *et al.*, 2020). It is a hierarchical security model designed to protect system assets (or objects) from unauthorised modification; which is to say it is designed to protect system integrity.

Biba model is based on the realisation that an entity with high integrity level is more reliable than a lower-ranked entity. In this model, subjects and objects are associated with integrity levels where subjects can modify objects only at a level equal to or below their integrity level (Liu *et al.*, 2017). The Biba model consists of the following access modes (Moe and Thwin, 2019):

a) Modify: This gives subjects the write privilege to objects;
b) Observe: This gives subjects the read privilege to objects;
c) Invoke: This permits subjects to communicate with other subjects; and
d) Execute: This allows a subject to execute an object.

Strengths of Biba Integrity Model: According to Toapanta *et al*. (2018) and Schinagl, Paans and Schoon (2016), the strengths of the Biba Model are:

a) The Biba model is simple and easy to implement;
b) It can easily be combined with the BLP model to provide a hybrid security model that can provide both confidentiality and integrity security;
c) Its implementation is intuitive and easily understood;
a) Biba model is a common commercial security model; and
b) The Biba model provides several different policies that can be selected based on need.

Problems with Biba Integrity Model: The problems with Biba model, according to Yadav and Shah (2015), include:

a) The model only solves the integrity problem without considering the confidentiality and availability;
b) There is no instruction to manage the access control and no method about how to distribute and change the classification level;
c) Biba model does not support the granting and revocation of authorisation; and
d) To use this model, all computers in the system must support the labelling of integrity for both subjects and objects.

Westmacott (2019) proposed the use of the Biba model, which is specifically designed to protect data integrity to solve the problem of online attack. With the use of the simple integrity property (read up), star integrity property (write down), and the discretionary security property, the model was proposed to be feasible and could prevent users from reading posts from users who were less identifiable irrespective of the intended recipient.

The proposed model was also stated to prevent users of low identifiability from sending posts to identifiable users, nevertheless, the drawback is that the model was not implemented, and performance evaluation was not made.

Liu *et al.* (2017) researched on the flexibility enhanced Biba integrity model using BTG strategy to secure operating systems. Although the traditional Biba integrity model can protect information integrity, it sometimes denies various access requests of subjects, thereby decreasing the availability of a system. Therefore, a mechanism that allowed exceptional access control was proposed using the BTG strategy to provide both an original Biba model used in normal situations and a mechanism used in emergencies. BTG is based upon a pre-staged emergency user accounts and allows emergency access to the system. The limitation of the study was that BTG mode was not open to all the subjects in the system.

2.7 Voting Model

The voting model applied in this research was adopted from Artificial Neural Network (ANN). ANN is a mathematical model that consists of an interconnected group of artificial neurons for modelling complex relationships between inputs and outputs. ANN can be perceived as a weighted directed graph in which artificial neurons serve as nodes with directed edges, which also serve as weights. The artificial neuron in ANN as shown in Figure 2.5, takes a set of inputs; x_1, ..., x_n, and multiplies with their respective weights to generate the output, O as represented in equation (2.7).

$$O = f\left(\sum_{i=1}^{n} w_i \cdot x_i\right) \tag{2.7}$$

where, w_i = weight;

i and n= integer; and

f = activation function.

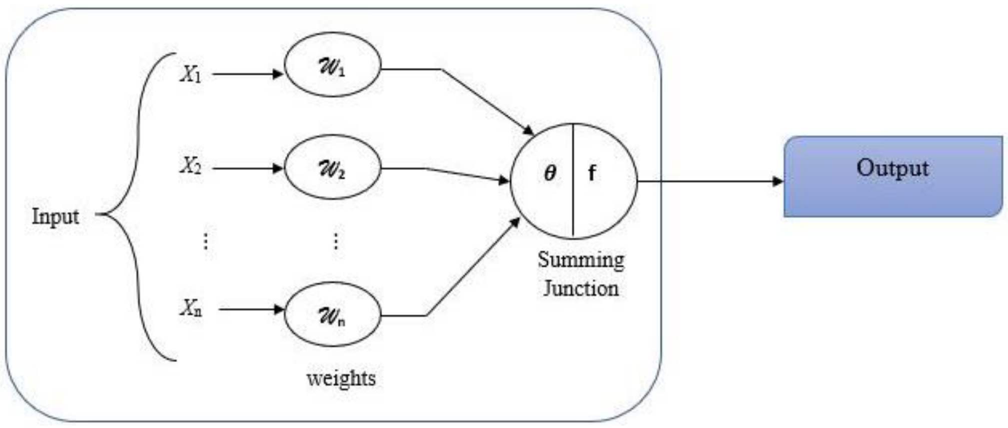


Figure 2.5 Artificial Neuron

2.8 Summary of Relevant Literature

Literature review presented in the areas of Analytic Hierarchy Process, Software Quality and Software Security models (Bell-LaPadula and Biba Integrity models) are summarised and presented in Table 2.8.

Table 2.8 Summary of Literature Review

Author(s), Year, Research Title	Objective	Methodology	Contribution to Knowledge	Limitation(s)
Analytic Hierarchy Process				
Kumar and Singh (2016), A Comprehensive Evaluation of Aspect-Oriented Software Quality (AOSQ) Model	To evaluate the Aspect-Oriented Software Quality model.	Analytic Hierarchy Process	The weights between attributes and sub-attributes helped in evaluating the model.	There were ranking irregularities.
Verma and Mehlawat (2017), multi-criteria optimisation model integrated with AHP for evaluation and selection of COTS components	To evaluate and select Commercial-off-the-shelf (COTS) components.	Analytic Hierarchy Process	Applying these weights as coefficients of an objective function in the proposed model helped to determine the best component.	The maximum number of alternatives to be compared at a time was small.
Yujun *et al*. (2019), Software Quality Risk Assessment Method for Information System	To calculate the weight and order of risk factors.	Analytic Hierarchy Process	The results showed that there was the need to pay more attention to the change of requirements and the development process.	Decision-makers found it difficult to convert from verbal to numeric scale.
Mahmudova and Jabrailova (2020), Development of an algorithm for selecting software.	To develop an algorithm to evaluate software functionality.	Analytic Hierarchy Process	The AHP method was applied for the first time to evaluate software functionality.	It was difficult to compute when the number of pair-wise comparisons became large.

Author(s), Year, Research Title	**Objective**	**Methodology**	**Contribution to Knowledge**	**Limitation(s)**
		Software Quality		
Kabir, Rehman and Majumdar (2016), An Analytical Study of Software Usability Factors.	To analyse ten famous quality models for a usability model.	McCall, Boehm, Shackel, FURPS, Nielsen, ISO 9242-11 and ISO 9126 models	An improved usability model that provides twelve usability factors was presented.	The research did not show the implementation of the model, hence, performance evaluation was not carried out.
Kassie and Singh (2020), A Study on Software Quality Factors and Metrics.	To propose a user's perspective-based software quality model.	Use of existing software quality models.	They identified the ten most important software quality attributes that are of importance to users.	The study did not cover a wide scope of quality attributes.
Parthasarathy *et al*. (2020), Quality Assessment of Standard and Customised COTS Products.	To assess the quality of standard COTS products.	ISO/IEC 9126 model.	A measurement of the quality attributes and sub-attributes of the ISO/IEC 9126 quality model was made.	The research did not address software availability problems.
Al-Nawaiseh, Helmy and Khalil (2020), A New Software Quality Model for Academic Information Systems: Case Study of E-Learning Systems.	To guide academic institutions that are in the process of building their E-learning systems to evaluate software attributes.	ISO/IEC 9126 quality model.	The research was able to build a standard approach that measures and evaluates the quality of AIS.	The proposed model failed to evaluate the importance of the quality attributes.

Author(s), Year, Research Title	**Objective**	**Methodology**	**Contribution to Knowledge**	**Limitation(s)**
Software Security Models				
Salman *et al*. (2017), Multi-Level Security for the 5G/IoT Ubiquitous Network	To present a multi-level security model.	The BLP model.	The model was proven to be secure by demonstrating the transition from one secure state to another secure state.	The model failed to define the security management tasks.
Saravanan and Umamakeswari (2020), Lattice Based Access Control for Protecting User Data in Cloud Environments	To protect patient's data on a secure cloud storage.	The BLP model.	The user authentication level was successfully implemented using the BLP model.	The user found it difficult accessing documents at higher security levels.
Liu *et al*. (2017), BTG-BIBA: A Flexibility-Enhanced Biba Model Using BTG Strategies	To secure operating systems.	Biba model with break-the-glass (BTG) strategy.	A mechanism that allowed exceptional access control was proposed using BTG strategy.	The limitation of the study was that, break the glass mode was not open to all the subjects in the system.
Westmacott (2019), Biba Security Model Inspired Social Media Security Controls	To protect the integrity of social media users	The Biba model.	The proposed model was proposed to be feasible enough to prevent reading posts from less identifiable senders irrespective of intended recipient.	Failed to address confidentiality issues.
Toapanta *et al*. (2018), Analysis of the Appropriate Security Models for a Distributed Architecture	The objective was to perform the analysis of security models.	The BLP and Biba security models.	The model provided an adequate security model to improve the confidentiality, integrity and authenticity of information.	The research did not show the implementation of the models; hence, evaluation was not made.

2.9 Research Gaps in Related Works

Researchers in Section 2.8.3 have applied software quality attributes to evaluate COTS, ERP systems, AIS, and web-based software but these works did not capture a higher scope of quality attributes. As a result, some vital software quality attributes were not addressed. Also, most of these works have focused on proposing quality models tailored towards specific project's needs (Galli, Chiclana, and Siewe, 2020), hence, a generic model that can suit all software projects is sought for. Additionally, the quality attributes addressed by these researchers have not been ranked to allow easy identification of the most important attributes to use for projects (Thamer, Mohammad and Ahmad (2013); Alanazi *et al*. (2019); Al-Nawaiseh, Helmy and Khalil (2020)). Furthermore, the researchers did not factor in all the quality attributes either directly or indirectly for assessing the quality of software.

Secondly, works by researchers in Section 2.8.2 are on BLP and Biba models for ensuring confidentiality and integrity. The core security goals according to Dorri *et al*. (2017) are confidentiality, integrity, and availability but most of these works have eliminated some of the core security goals (Zhu *et al*. (2016); Liu *et al*. (2017)).

Moreover, most of the works have not been implemented (Kabir, Rehman and Majumdar (2016); Toapanta *et al*. (2018)). As a result, evaluation of these works has not been made to assess the performance of the models.

2.10 Justification of Methods Used

2.10.1 The Proposed Quality Model

Software quality models provide the necessary basis to determine the quality of software based on attributes such as flexibility, maintainability, reliability, testability, efficiency, security, portability, understandability, integrity, functionality, usability, interoperability, reusability, and robustness. Although there are other existing quality attributes, these attributes were chosen based on their distinct characteristics and wide application by researchers. A comparison between ten (10) software quality models which include Kitchenham and Pickard model, McCall model, FURPS model, Georgiadou model, Boehm's model, Glib model, Ghezzi model, ISO model, Dromey's model, and Jamwal model was drawn.

2.10.2 The Secured Model

Over the years, many models have been developed using BLP and Biba models to address confidentiality and integrity issues. The Biba model addresses the problem with the star property of the Bell-LaPadula model, which does not restrict a subject from writing to a more trusted object. Hence, the hybrid model consisting of BLP and Biba models will complement and address the shortcomings of each other.

2.10.3 AHP Technique

The AHP technique was applied in this research because it is a robust decision-making tool. It is also flexible in dealing with complex decision problems and uses a multi-level hierarchical structure of objective or goal at the top level, criteria or attributes at the second level, and alternatives at the third level.

CHAPTER 3

SYSTEM DESIGN

3.1 Introduction

The methodology for achieving the stated objectives is divided into four (4) phases. Phase one categorises the software quality attributes into main attributes and sub-attributes. Phase two employs the use of AHP to carry out a multi-criteria decision-making analysis of the software quality attributes. Phase three involves the actual design and implementation of the software quality attributes using the voting method. Finally, phase four covers the design and implementation of the access control system for the overall quality assurance software.

In order to achieve the design of the proposed software quality model, individual standard quality attributes were identified, and thereafter, combined using a voting model. For the design of the access control model, Bell-LaPadula was used for the login process while Biba model was used at the account registration stage.

3.2 Quality Attributes and Sub-Attributes

Twenty-four (24) software quality attributes were initially sampled using purposive sampling technique from the following ten (10) standard and well-known software quality models: Georgiadou's, Dromey's, Glib's, ISO 9126, McCall's, Kitchenham and Pickard's, FURPS, Ghezzi's, Boehm's, and Jamwal's models. The quality attributes are Interoperability, Non-Repudiation, Efficiency, Security, Cost, Supportability, Flexibility, Correctness, Portability, Adaptability, Integrity, Understandability, Testability, Reusability, Maintainability, Reliability, Usability, Functionality, Performance, Availability, Extensibility, Confidentiality, Accuracy, and Robustness. Review of related works showed that some of the quality attributes had similar functionality as others and were grouped into main and sub-attributes.

3.3 Quality Assurance Model

The quality assurance model consists of eleven (11) main attributes and thirteen (13) sub-attributes as shown in Figure 3.1. Maintainability has Flexibility, Extensibility and Supportability as its sub-attributes. Security is also seen to have Integrity, Confidentiality and Non-Repudiation as its sub-attributes. Functionality is seen to have Correctness and

Interoperability as its sub-attributes. Also, Reliability has Robustness and Accuracy as the sub-attributes. Usability has Understandability as its sub-attribute while Efficiency has Performance as its sub-attribute. Lastly, Portability has Adaptability as its sub-attribute. Reusability, Testability, Availability and Cost have no sub-attributes.

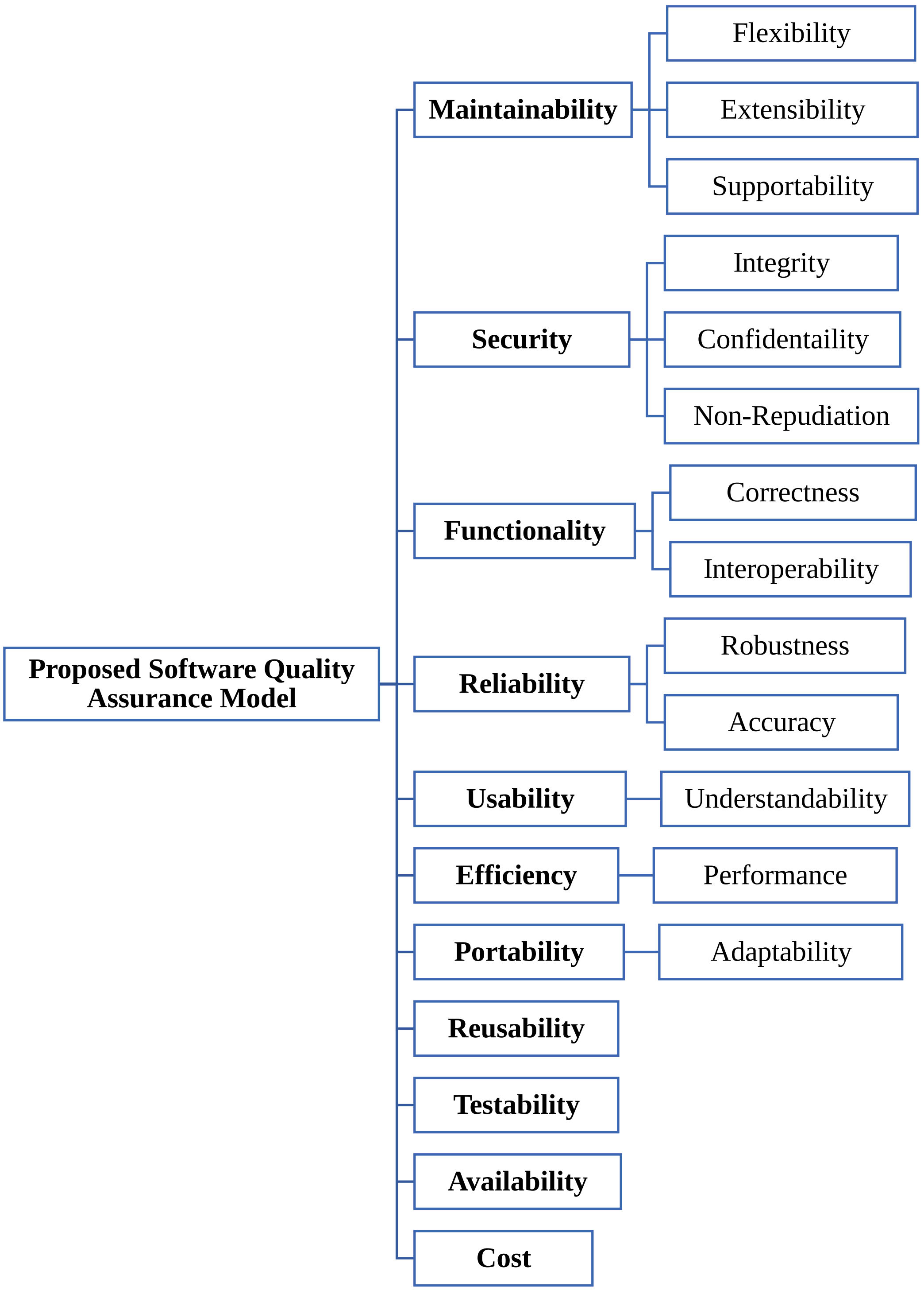


Figure 3.1 Hierarchical Structure of the Proposed Quality Assurance Model

3.4 Assessment of Software Quality Attributes

The research used the AHP technique to rank the software quality attributes which will be used in the development of the quality model.

The ranking was made by using the attributes selected under Section 3.3, namely, Testability (T), Security (S), Usability (U), Cost (C_o), Efficiency (E), Reusability (R_e), Maintainability (M), Availability (A), Reliability (R), Functionality (F_n), and Portability (P), and three (3) alternatives, i.e., Doubles up as Sub-attribute, Has sub-attributes, and Mostly addressed. This data was used to develop an ordered structure with the goal at the top level, the attributes at the second level, and the alternatives at the third level as shown in Figure 3.2. The obtained hierarchical structure was synthesised to determine the relative importance of the different attributes to the goal. This is done using a pair-wise comparison matrix with the help of a scale of relative importance as shown in Table 3.1.

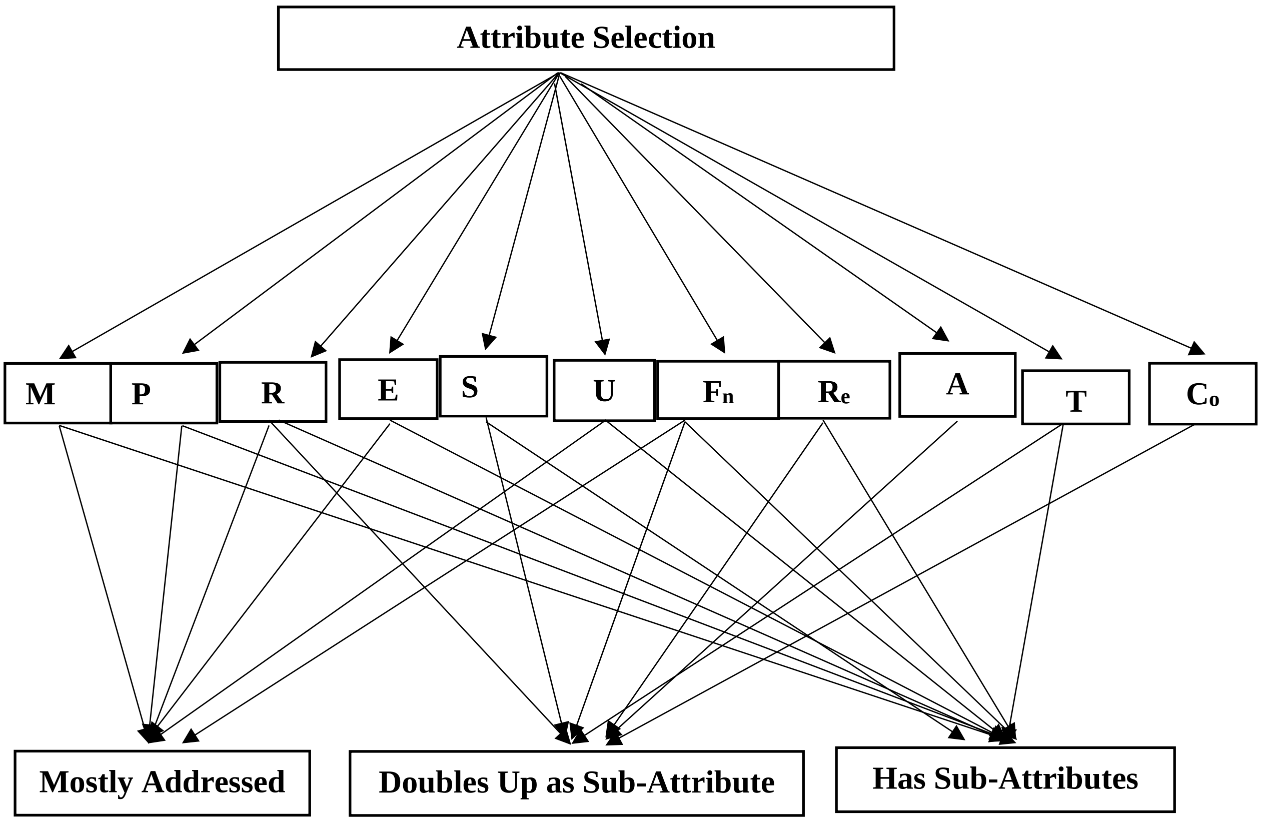


Figure 3.2 Hierarchical Structure of Software Quality Attributes

Table 3.1 Scale of Comparison

Scale of Importance	**Degree of Preference**
1	Equal Importance
3	Moderate Importance
5	Strong Importance
7	Very Strong Importance
9	Extreme Importance
2,4,6,8	Intermediate Values
1/3, 1/5, 1/7, 1/9	Values for Inverse Comparison

(Source: Saaty, 2008)

The number of comparisons is a combination of the number of things to be compared as shown in Table 3.2.

Table 3.2 Number of Comparisons

Number of Things	1	2	3	4	5	6	7	n
Number of Comparisons	0	1	3	6	10	15	21	$\frac{n(n-1)}{2}$

A primary questionnaire was designed, as shown in Table 3.3 and given to thirty (30) experts in Ghana and Nigeria from the fields of Cybersecurity, Software Programming, Software Development and Software Engineering to complete.

Table 3.3 Questionnaire Given to Expert

Attributes	**M**	**T**	**R**	**P**	**A**	**E**	**F_n**	**R_e**	**S**	**U**	**C_o**
M	1										
T		1									
R			1								
P				1							
A					1						
E						1					
F_n							1				
R_e								1			
S									1		
U										1	
C_o											1

After completion of the questionnaire by the experts, a matrix comprising of the eleven (11) attributes was generated. The matrix was filled by asking the importance of one attribute relative to the other. Since there are eleven (11) comparisons, an 11 × 11 matrix was generated. The diagonal of the matrix is always 1 and the upper triangle of the matrix is filled using the following rules according to Saaty (1977):

i. If the judgement value is on the left side of 1, we write the actual judgement value; and
ii. If the judgement value is on the right side of 1, we write the reciprocal judgement value.

To fill the lower triangular matrix, the reciprocal values of the upper diagonal are used. If c_{ij} is the element of row *i* column *j* of the matrix, then the lower diagonal is filled using

$$c_{ji} = \frac{1}{c_{ij}} \tag{3.1}$$

The comparison matrix is generated as

$$\begin{bmatrix} c_{11} & \cdots & c_{1\,24} \\ \vdots & \ddots & \vdots \\ c_{5\,1} & \cdots & c_{5\,24} \end{bmatrix} \tag{3.2}$$

The comparison matrix is normalised to calculate the consistency vector (relative weight). The relative weight is given by the eigenvector, W, which is the largest eigenvalue, λ_{max}. This was calculated from the comparison matrix by multiplying the pair-wise matrix by the weight vector and the sum of the row entries were divided by the corresponding criterion weight.

To evaluate the consistency of one’s judgement, the Consistency Index (CI) is calculated as shown, in equation (3.3).

$$\mathrm{CI} = \frac{\lambda_{\max} - \mathrm{n}}{\mathrm{n} - 1} \tag{3.3}$$

where, n = order of the matrix.

The Consistency Ratio (CR) is also calculated using equation (3.4).

$$\text{CR} = \frac{CI}{RI} \tag{3.4}$$

where, RI = Random Index and it is as shown in Table 3.4.

Table 3.4 Number of Comparisons with the corresponding RI value

n	1	2	3	4	5	6	7	8	9	10	11
RI	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.52

If $\text{CR} \leq 0.1$, the judgement is seen to be acceptable, else the judgement is to be re-examined.

The process is repeated to analyse the alternatives as well. The weighted matrix of the three (3) alternatives is multiplied with the weighted matrix of the attributes or criteria.

The structure of the final decision matrix is shown as:

$$\begin{array}{cccccc} & W_1 & W_2 & W_3 & \cdots & W_n \\ A_1 & a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ A_2 & a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ A_3 & a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ A_m & a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{array}$$

where, W_1 to W_n = Criteria;

A_1 to A_m = Alternatives; and

a_{mn} = number in row m and column n.

The overall consistency ratio, $\overline{CR}$, was calculated as shown in equation (3.5) by summing up the weighted consistency index, w_iCI_i, in the nominator and the weighted random consistency index, w_iRI_i, in the denominator.

$$\overline{CR} = \frac{\sum_i w_i CI_i}{\sum_i w_i RI_i} \tag{3.5}$$

3.5 Mathematical Models for the Secured Quality Assurance Model

The twenty-four (24) attributes from Section 3.2 were reduced to eleven (11) because of similarities of the attributes. Thereafter, AHP was used to carry out a multi-criteria decision-making analysis on the attributes.

The attributes used to implement the software quality assurance model are Reusability, Testability, Reliability, Availability, Efficiency, Maintainability, Usability, Functionality, Cost, Security, and Portability. Confidentiality and Integrity were used to implement the access control security model which will secure the overall quality assurance model.

3.5.1 Mathematical Model for the Access Control Security Model

The attributes for the access control security model were mathematically modelled using the BLP and Biba models.

Confidentiality

This refers to the state of denying unauthorised people the privilege to assess information. The **Star property** (* - property) of the BLP model was used to ensure software confidentiality. It denies subjects the write privilege to objects at lower security levels and is written mathematically using equation (3.6).

$$c_s \leq c_o \tag{3.6}$$

where, c_o = state of the object; and

c_s = state of the subject.

Integrity

This refers to the state of preventing unauthorised people from destroying or altering data. The simple integrity rule of the Biba model was used to ensure integrity. It states that subjects at a given level are denied read privilege of data at lower sensitive levels and is written mathematically as:

$$i_s \leq i_o \tag{3.7}$$

where, i_s = integrity level of the subject; and

i_0 = integrity level of the object.

Confusion Matrix for Access Control Model

The confusion matrix is used to evaluate the performance of a model. The confusion matrix of the access control model was calculated to evaluate user data points that were rightly predicted as the True Positive (TP) values, True Negative (TN) values, False Positive (FP) values, and False Negative (FN) values as shown in Table 3.5.

Table 3.5 Confusion Matrix

Results (n=100)	**Access Granted**	**Access Denied**
Access Granted	TP	FP
Access Denied	FN	TN

(Source: Zeng, 2019)

The accuracy of predicted values was evaluated using equation (3.8).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{3.8}$$

In order to evaluate the number of correctly "granted user access" that turned out to be true, the precision was found using equation (3.9).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.9}$$

To evaluate the "granted user access" cases that were correctly predicted by the model, the recall was calculated using equation (3.10).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.10}$$

F1 score was also calculated using equation (3.11) to find a balance between Precision and Recall.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.11}$$

3.5.2 Mathematical Model for the Quality Assurance Model

The quality assurance model was modelled using the eleven (11) software quality attributes that were ranked by the AHP. Their mathematical models were used to implement each of the attributes.

Maintainability

Maintainability shows how easily a system can be repaired once it encounters an error. The higher a system is maintained, the lower the mean time it takes to repair. The time it takes to repair includes the repair process time and return to service time. This is together encapsulated in the Mean Time To Recover (MTTR). Improving an application's MTTR improves its maintainability. Maintainability, M, can be expressed mathematically in equation (3.12).

$$\mathrm{M} = \mathrm{MTTR} \tag{3.12}$$

MTTR is expressed in terms of the Total downtime, T_D, and the number of failures, F_N, a web application encounters during operation. This is expressed mathematically in equation (3.13).

$$\mathrm{MTTR} = \frac{\text{Total Downtime}}{\text{Number of Failures}} \tag{3.13}$$

Therefore, Maintainability of a web application software is calculated using equation 3.14.

$$\mathrm{M} = \frac{T_D}{F_N} \tag{3.14}$$

Testability

It is necessary to verify the requirements of software. Testing takes up a lot of time and effort in software development as it is used to determine whether the desired user requirements have been met, whether the software functions correctly, and so on. Web application software needs to be tested before it is made publicly accessible. Testability is a function of correctness, reliability, and the time taken to test the web application. Testability, T, can be expressed mathematically using equation (3.15).

$$T = \lim_{x \to \infty} \left(1 + \frac{1}{x}\right) . \, e^{(-\lambda . t)} \, . \, \varepsilon_r \tag{3.15}$$

where, t = time taken to test the web application;

λ = failure rate of the system;

ε_r = error rate; and

x = number of constraints being considered.

The number of constraints used may include memory usage and throughput. The failure rate of the software was evaluated by conducting a load test where 500 concurrent users were simulated. The value of the failure rate was attained from the number of failed connections

(connections that were refused by the software) and the number of failed hits (failed attempts to retrieve data).

Reliability

Reliability, R, defines how a system works under specific conditions over time. To test for the Reliability of software, it is necessary to test it under ways it is likely to encounter failure. This is done by conducting high accelerated life tests where the software is put under stress to determine the software's limitations and test its error handling capabilities under extremely heavy conditions. The failure rate of the software is determined from the number of failed hits and the number of failed connections. The reliability of a system varies exponentially as a function of time. It can be shown in equations (3.16).

$$R = e^{(-\lambda . t)} \tag{3.16}$$

where, t = period the software product was put to use; and

λ = failure rate.

Software reliability is a probabilistic feature and ranges between 0 and 1. It increases when bugs are removed from the software.

Efficiency

Efficiency, E, is the ability of software to offer the right performance relative to the given and used resources. An efficient software fulfils its purpose without resource wastage. The efficiency of a web-based application may be calculated using throughput and bandwidth.

Throughput is the number of items processed per unit time, such as bits transmitted per second, HTTP operations per day, or millions of instructions per second (MIPS). It is used to check how many requests a web-based application will be able to process per second, per minute, or hour. Throughput is an important metric in evaluating web-based applications because it is used to determine how much bandwidth is required to handle a load of both concurrent users and website requests.

To calculate for Throughput of a software, equation (3.17) is used:

$$\text{Throughput} = n_i \tag{3.17}$$

where, n_i = number of requests being sent to the servers per unit time;

Bandwidth is the measurement of the amount of data a software uses during a specific time frame. For web-based applications, the amount of traffic and the number of resources (images, files, graphics, and others) affect bandwidth. Bandwidth restrictions have significant practical implications because, when exceeded, they can greatly increase end-user frustration and seriously degrade interaction with a web application.

The efficiency of a web-based application may be expressed mathematically in equation (3.18).

$$E = \frac{ST}{SB} . 100 \tag{3.18}$$

where, ST = Throughput; and

SB = Bandwidth.

Availability

Availability refers to the ability of users to access and use a web-based application. An available software is accessible and usable as expected by the user. When speaking about availability, we often refer to the ratio of the available time to the total time. It emphasizes the repair time and restart time of the web application. A web application's availability is typically calculated as a percentage for a given period. It can be expressed in terms of Mean Time Between Failure (MTBF) and MTTR. MTBF and MTTR are calculated using equations (3.19) and (3.13) respectively.

$$\text{MTBF} = \frac{\text{Number of Operational hours}}{\text{Number of Failures}} \tag{3.19}$$

Availability can be expressed mathematically using equations (3.20) and (3.21).

$$A = \frac{Z}{Z + Y} \tag{3.20}$$

where, Z = Mean Time Between Failure; and

Y = Mean Time to Recover.

Availability reaches 100% when there is an instant repair once failure is encountered and the MTTR approaches 0. It is also represented in terms of operational hours, O, and downtime, T_D.

$$A = \frac{O}{O + T_D} \quad (3.21)$$

Usability

Software usability assessment is important because it aids in evaluating performance and user fulfilment of a product. Immediately users face difficulty in website navigation, they tend to look for other websites with similar functionalities. Website usability evaluation was performed based on a survey approach using the ISO 9126, ISO 9241-11 and Nielsen usability models (Nielsen, 2003; Nielsen, 2012) and survey questions that were modelled from the System Usability Scale (SUS) and Post-Study Survey Usability Questionnaire (PSSUQ). The survey approach gathered answers on learnability, navigation, effectiveness, clarity of information, understandability, and others from the respondents.

Questionnaire-based Usability Evaluation: The usability evaluation was performed by administering a questionnaire to users. This questionnaire, as shown in Table 3.6, was completed using a scale defined from 1 to 5, with 1 as Strongly Disagree, 2 as Disagree, 3 as Undecided, 4 as Agree, and 5 as Strongly Agree.

Table 3.6 Survey Questions and Scale Used

No.	**Question**	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
		5	**4**	**3**	**2**	**1**
1	There is an easy navigation within the web application					
2	The information I needed were readily available					
3	There was a clear organisation of information					
4	The website had a pleasant interface					
5	There were useful images on the website					
6	There was an orderly presentation of content					
7	The size of web controls was appropriate					
8	The website had less loading time					

Table 3.6 Survey Questions and Scale Used (cont'd)

No.	Question	Strongly Agree	Agree	Undecided	Disagree	Strongly Disagree
		5	4	3	2	1
9	The website has all the needed functions					
10	The website is satisfactory					

The results from the questionnaire were evaluated for Reliability using Cronbach Alpha mathematical method which is expressed in equation (3.22).

$$\alpha = N\bar{c}/(\bar{v} + (N - 1) * \bar{c}) \qquad (3.22)$$

where, α = Cronbach Alpha;

N = Number of items;

$\bar{c}$ = Covariance between the items; and

$\bar{v}$ = average variance.

The scores from the questionnaire were implemented in IBM SPSS Statistics 26.0 software. The value for Cronbach Alpha ranges between 0 and 1 and signifies high reliability when the value is closer to 1 as shown in Table 3.7.

Table 3.7 Relationship Between Cronbach Alpha's Score and Reliability

Cronbach Alpha's Score	Level of Reliability
$\alpha \geq 0.9$	Excellent (Very Reliable)
$0.9 > \alpha \geq 0.8$	Good (Reliable)
$0.8 > \alpha \geq 0.7$	Acceptable (Quite Reliable)
$0.7 > \alpha \geq 0.6$	Questionable (Rather Reliable)
$0.6 > \alpha \geq 0.5$	Poor (Less Reliable)

(Source: Polat *et al.*, 2017)

The usability, U, of the web applications were individually calculated using equation (3.23) as:

$$U = \frac{TS}{MS}.100 \qquad (3.23)$$

where, TS = Total Score; and

MS = Maximum Score.

The usability value was graded after calculation using the SUS as shown in Table 3.8.

Table 3.8 SUS Score and Grade

SUS Score	Grade	Rating
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Okay
51 - 68	D	Poor
< 51	F	Awful

(Source: Derisma, 2020)

Reusability

Reusability can be measured by the time it takes a software to deliver (delivery time) and the correctness of the software (Ogundele, 2018). Delivery time is the expected time for the software to return the same results under the same conditions after several usages. The higher the delivery time value, the better the reusability of software components. Delivery time may be expressed in terms of MTBF and MTTR in equation 3.24 as:

$$\text{Delivery Time} = \frac{\text{MTBF}}{\text{MTTR}} \tag{3.24}$$

Reusability, R_e, may also be expressed using equations (3.25) and (3.26).

$$R_e = \lim_{x \to \infty} \left(1 + \frac{1}{x}\right) + \text{Delivery Time} \tag{3.25}$$

$$R_e = \lim_{x \to \infty} \left(1 + \frac{1}{x}\right) + \frac{\text{MTBF}}{\text{MTTR}} \tag{3.26}$$

When the value of correctness converges to 1, the equation as shown in (3.27) and (3.28) become;

$$R_e = 1 + \frac{\text{MTBF}}{\text{MTTR}} \tag{3.27}$$

$$\text{R}_e = 1 + \frac{\text{O}_p}{\text{T}_D} \tag{3.28}$$

where, O_p = Operational Time; and

T_D = Total Downtime.

Functionality

This is the ability of software to perform the tasks for which it was intended. It has a direct relationship with the components of the software and may be expressed mathematically using the working and not working functions in the software as shown in equation (3.29).

$$F_n = \left(1 - \frac{A_o}{B_o}\right) . 100 \tag{3.29}$$

where, F_n = Functionality;

A_o = number of functions that are not working correctly; and

B_o = number of functions that are working correctly.

The functions used in the test include submission of forms, search box working correctly, live chat feature working correctly, social media tabs redirecting correctly, internal links functioning, site map aiding in user navigation, functioning print page feature, correctly working events calendar, and others.

Portability

This is an approach that depicts the ease with which a software or an application can run across various computing platforms. Portability tests can be done across various hardware platforms, operating systems, or web browsers. This helps to find out the ease with which a software component from one computing environment can be used in another environment. Portability, P, is expressed as a form of accelerated motion as shown in equation (3.30).

$$P = \frac{1}{2}(at_o{}^2 + vt_o + P_o) \tag{3.30}$$

where, a = acceleration of the software across various platforms;

t_o = time taken to move across various platforms;

v = speed of moving across various platforms; and

P_o = rate of transfer across various platforms.

Acceleration was used to evaluate the rate of change of the software's speed across multiple web browsers, speed was used to evaluate the rate at which the software opens in a web browser, time evaluates the average time it takes the software to open on multiple browsers while the rate of transfer is the total time it takes the software to open on multiple web browsers. The web browsers used for the test were Google Chrome version 89.0, Mozilla Firefox version 87.0, Microsoft Edge version 89.0, and Safari version 5.1.

Security

The security of the web applications was categorised under the Open Web Application Security Problem (OWASP) top ten (10) security vulnerability model version 2017. Security, S, may be expressed as shown in equation (3.31).

$$S = \sum_{i=1}^{10} A_i \qquad (3.31)$$

where, A = Security Vulnerability; and

i = Vulnerability Level.

The OWASP top ten (10) security vulnerability are classified under Code Injection attack (A_1), Broken Authentication and Session Management (A_2), Cross-Site Scripting attack (XSS) (A_3), Insecure Direct Object References (A_4), Security Misconfiguration (A_5), Sensitive Data Exposure attack (A_6), Missing Function Level Access Control (A_7), Cross-Site Request Forgery (CSRF) (A_8), Using Components with Known Vulnerabilities (A_9) and Unvalidated Redirects and Forwards (A_{10}). The Security Vulnerability Level and the corresponding scores are shown in Table 3.9.

Table 3.9 Vulnerability Level and Score

Security Vulnerability Level	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
Score	10	10	10	10	10	10	10	10	10	10

Cost

Software Cost is the amount of money paid for software development. It is estimated using function points (FP), source lines of codes (SLOC), and labour used. Functional point parameters are calculated using External Interface Files (EIF), External Inputs (EI), Internal Logic Files (ILF), External Outputs (EO), and External Inquiries (EQ). The functional point parameter and the weight of complexity are shown in Table 3.10.

Table 3.10 Functional Point parameter and Weight of Complexity

Functional Point Parameter (FPP)	**Weight of Complexity (WC)**		
	Low	**Average**	**High**
External Inputs (EI)	3	4	6
External Outputs (EO)	4	5	7
External Inquiries (EQ)	3	4	6
Internal Logic Files (ILF)	7	10	15
External Interface Files (EIF)	5	7	10

(Source: Hana, Abeer and Hana, 2019)

Function Point, FP, was calculated using equation (3.32).

$$\text{FP} = \text{UFP} \times \text{CAF} \tag{3.32}$$

where, UFP = Unadjusted Functional Point; and

CAF = Complexity Adjustment Factor.

To calculate UFP, the values from the functional point parameter and their corresponding weight of complexities are summed up. This is expressed mathematically in equation (3.33).

$$\text{UFP} = \sum (\text{FPP} \times \text{WC}) \tag{3.33}$$

where, FPP = function point parameter; and

WC = weight of complexity.

CAF is also calculated using equation (3.34) as:

$$\text{CAF} = 0.65 + (0.01 \times \sum \text{F}_i) \tag{3.34}$$

where, F_i = value adjustment factor based on responses to questions in Table 3.11.

The questionnaire was filled using a scale defined from 1 to 5, with 1 as Incidental, 2 as Moderate, 3 as Average, 4 as Significant, and 5 as Essential.

Table 3.11 Questions for the value adjustment factor

No.	**Question**	**Essential**	**Significant**	**Average**	**Moderate**	**Incidental**
		5	**4**	**3**	**2**	**1**
1	Data Communication					
2	The software requires distributed data processing					
3	What is the rate of performance					
4	The software has a heavily used configuration					
5	It has a transaction role					
6	Allows data entry					
7	Requires end-user efficiency					
8	Allows online update					
9	The software uses complex processing					
10	The components of the software are reusable					
11	There should be ease of installation					
12	Allows operational use					
13	The software uses multiple sites					
14	It facilitates change					

SLOC was calculated using function points and programming language used as shown in equation (3.35). The programming language used was evaluated using Table 3.12.

Table 3.12 Programming Language and Score Points for AVC

Programming Language	Score Points for Average Lines of Codes (AVC)
JAVA	53
C	97
C++	50
COBOL	61
C#	54
HTML	34
.NET	57
PYTHON	55
PHP	52

(Source: Duke and Obidinnu, 2010)

$$\text{SLOC} = \text{FP} \times \text{AVC} \tag{3.35}$$

The effort required to develop the software is calculated in terms of KLOC in equation (3.43) as:

$$Effort = a \times (KLOC)\hat{}b \tag{3.36}$$

where, KLOC = lines of codes in thousands;

a and b = Factors.

Table 3.13 Mode and Factors

Mode	a	b
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

(Source: Balaji, Shivakumar and Ananth, 2013)

Software cost estimate is calculated using Effort, E, and Labour, L, used as shown in equation 3.37.

$$C_o = E \times L \tag{3.37}$$

3.6 Voting Method

The voting method multiplies values from the score of attributes from the software quality assurance model with the scores attained from the AHP technique.

After the multiplication, the scores are summed up at the summing junction. The value at the summing junction lies between 0 and 100%. This is then outputted and displayed as the overall quality assurance evaluation of the application. Figure 3.3 shows the voting method used. The voting method was performed using equation (3.38).

$$\text{Output} = \sum_{n=1}^{11} (\text{SQA} \times w_{k1}) \tag{3.38}$$

where, SQA= Software Quality Attribute

w_{k1} = Weight generated from AHP

n = number of quality attributes

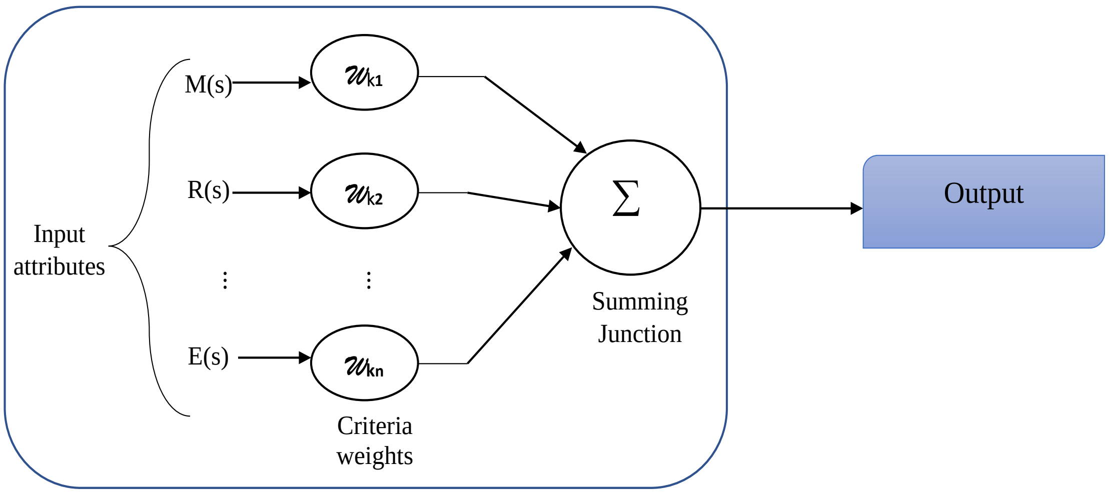
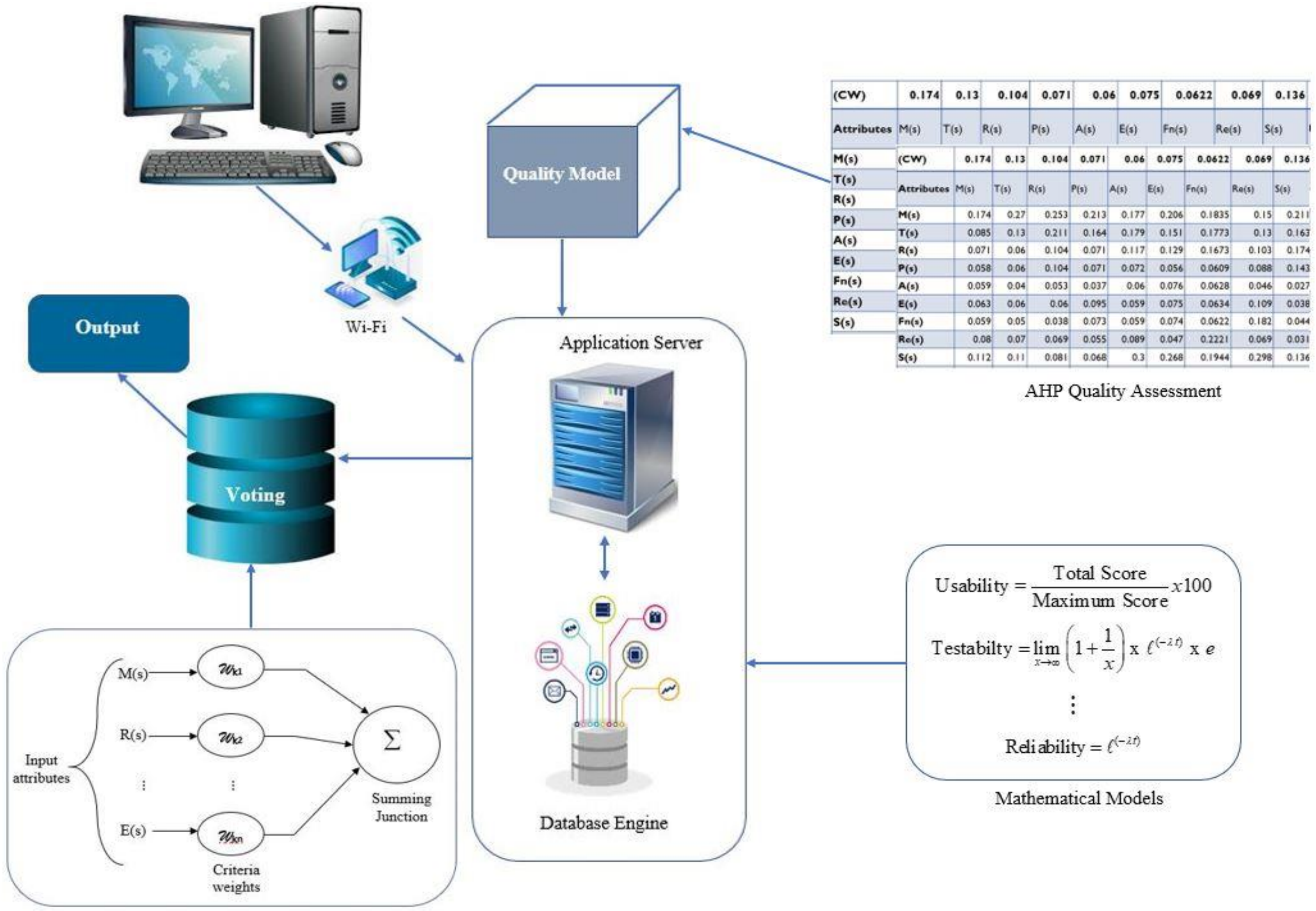


Figure 3.3 Voting Method Technique

3.7 Architecture of the Proposed Model

The architecture of the proposed model depicts the organisation and structure of the whole development model. It also shows how the model will operate as shown in Figure 3.4. The model contains a software quality model container that houses the eleven (11) quality attributes for the evaluation process. It also contains each attribute's corresponding criteria weights which were evaluated from the AHP analysis made. There is an application server to provide an environment to run the software and a database engine that contains the mathematical models for evaluating each software quality attribute. Finally, the model contains a voting system where the score of quality attribute attained by the evaluated web-based application software is multiplied with the corresponding AHP criteria weight, which lies between 0 and 100. The overall software quality assurance score is outputted and ranges between 0% and 100%. The higher the score from the software quality evaluation, the higher the quality of the web application.



(CW)	0.174	0.13	0.104	0.071	0.06	0.075	0.0622	0.069	0.136
Attributes	M(s)	T(s)	R(s)	P(s)	A(s)	E(s)	Fn(s)	Re(s)	S(s)
M(s)	0.174	0.27	0.253	0.213	0.177	0.206	0.1835	0.15	0.211
T(s)	0.085	0.13	0.211	0.164	0.179	0.151	0.1773	0.13	0.163
R(s)	0.071	0.06	0.104	0.071	0.117	0.129	0.1673	0.103	0.174
P(s)	0.058	0.06	0.104	0.071	0.072	0.056	0.0609	0.088	0.143
A(s)	0.059	0.04	0.053	0.037	0.06	0.076	0.0628	0.046	0.027
E(s)	0.063	0.06	0.06	0.095	0.059	0.075	0.0634	0.109	0.038
Fn(s)	0.059	0.05	0.038	0.073	0.059	0.074	0.0622	0.182	0.044
Re(s)	0.08	0.07	0.069	0.055	0.089	0.047	0.2221	0.069	0.031
S(s)	0.112	0.11	0.081	0.068	0.3	0.268	0.1944	0.298	0.136

$$\text{Usability} = \frac{\text{Total Score}}{\text{Maximum Score}} x100$$

$$\text{Testabilty} = \lim_{x\to\infty}\left(1+\frac{1}{x}\right) \text{x } \ell^{(-\lambda t)} \text{ x } e$$

$$\vdots$$

$$\text{Reliability} = \ell^{(-\lambda t)}$$

Figure 3.4 Architecture of the Proposed Model

3.8 Flow Diagram of the Proposed System

The system flow diagram, as shown in Figure 3.5, enables visualisation of the process flow in the model. The proposed model starts with a login process, which moves to the authorisation stage. Once the user is authorised, the dashboard is displayed where results for Usability are entered since it was done based on a survey approach. A request is then sent to the backend where Availability, Reliability, Reusability, Maintainability, Portability, Testability, Functionality, Cost, Efficiency, and Security tests are performed. The results move to the collation stage and are displayed for each of the quality attributes on the dashboard.

The voting method is carried out by multiplying the criteria weights from the AHP technique with the scores generated by each quality attribute in the software quality assurance model. The overall score from the voting method varies between 0 and 100% and shows the percentage of software quality for each evaluated web application.

The user is finally given the option to run a new test or end the process.

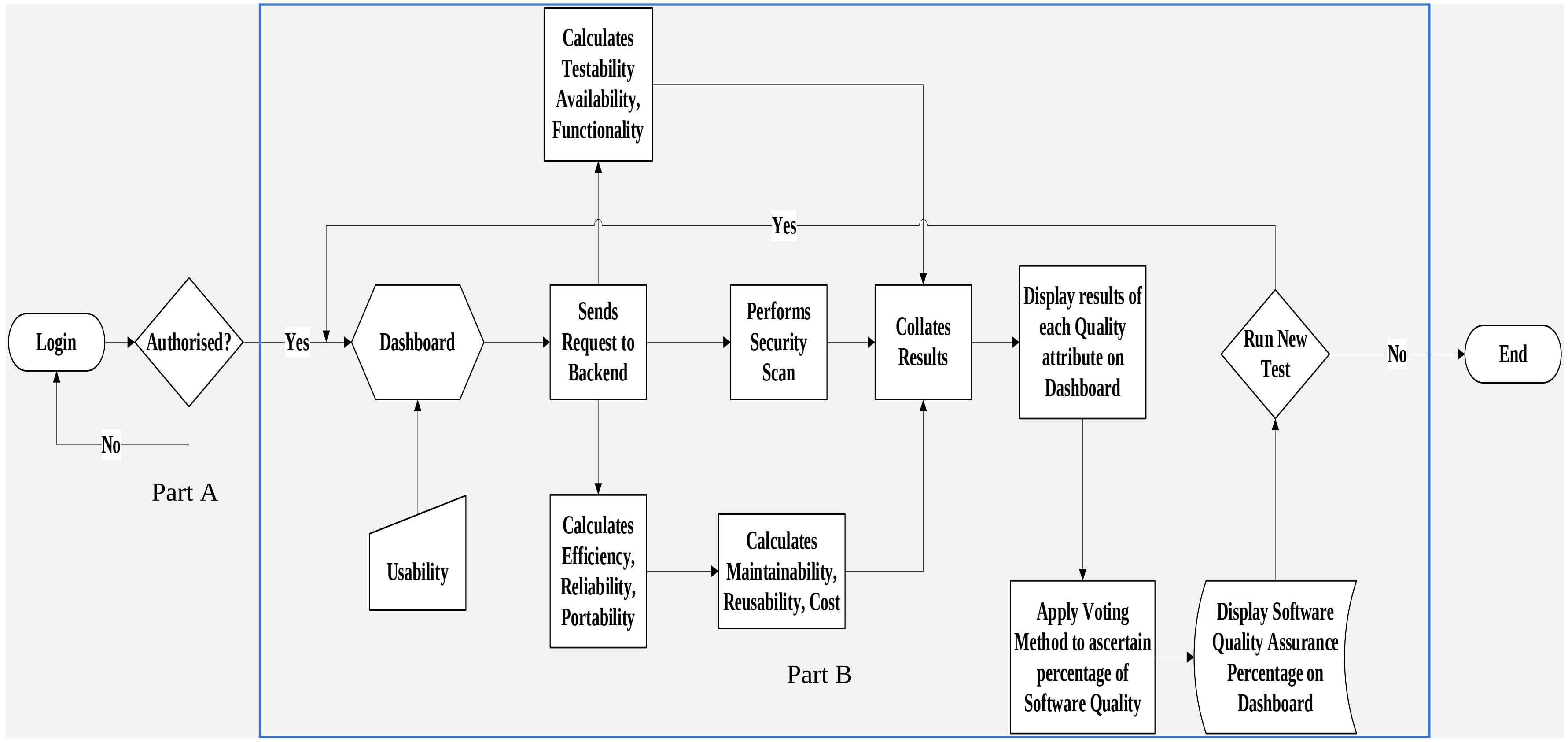

Login
Authorised?
Yes
No
Part A
Dashboard
Usability
Calculates Testability Availability, Functionality
Yes
Sends Request to Backend
Calculates Efficiency, Reliability, Portability
Performs Security Scan
Calculates Maintainability, Reusability, Cost
Collates Results
Part B
Display results of each Quality attribute on Dashboard
Apply Voting Method to ascertain percentage of Software Quality
Display Software Quality Assurance Percentage on Dashboard
Run New Test
No
End

Figure 3.5 Flow Diagram of the Proposed Model

The model also has an access control model that performs confidentiality check at the system login stage and integrity check at the account registration stage. A process flow diagram of the system login stage is shown in Figure 3.6.

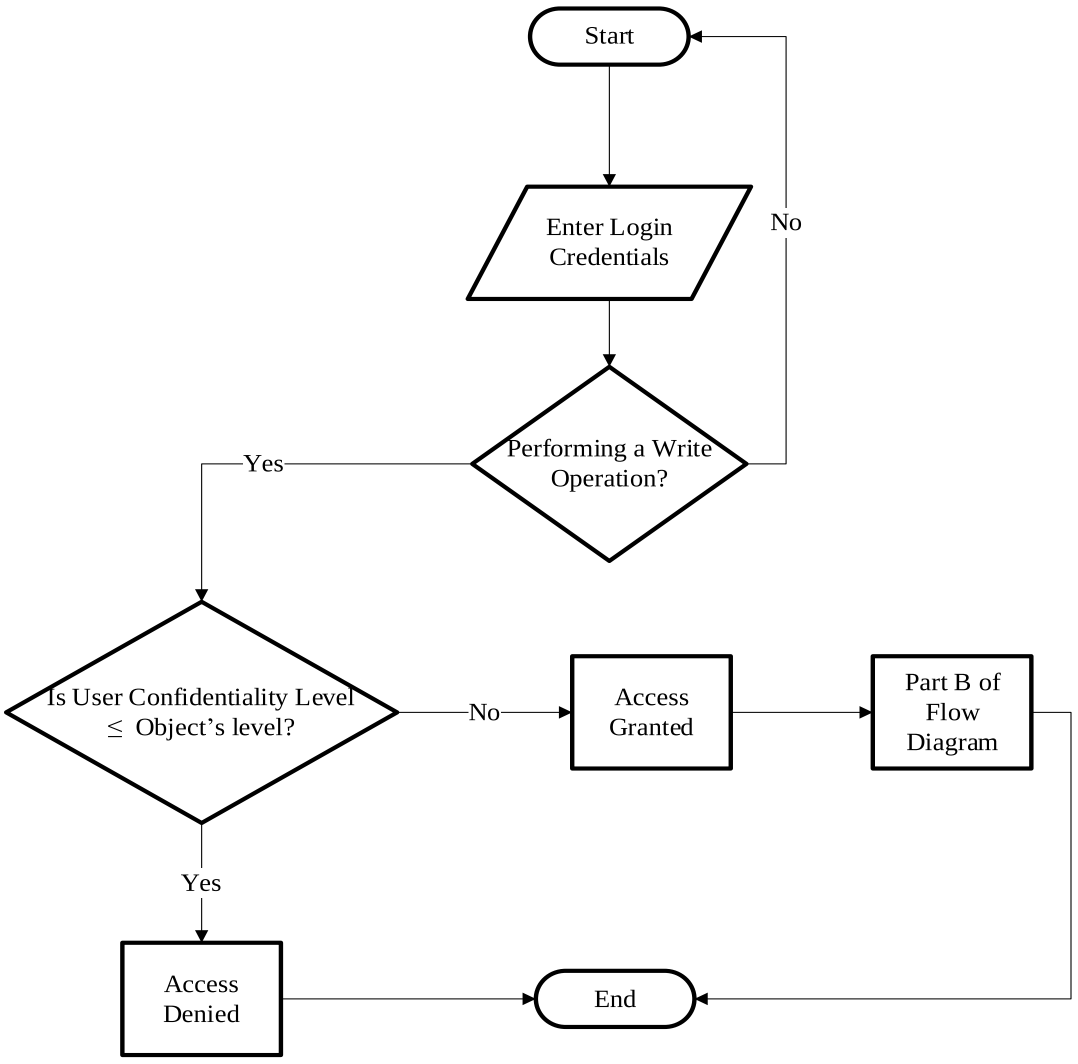


Figure 3.6 Flow Diagram of the System Login

The process is started for the user to enter the login credentials. It moves to the decision-making process where the system asks whether the user wants to perform a write operation; if yes, the user and object confidentiality levels are checked, else, it goes back to start the process. On checking the confidentiality level, if the user confidentiality level is less than or equal to the object's confidentiality level, the user is denied access to the system, else, the user is granted access to the software quality assurance model.

The account registration page performs an integrity check at the account activation stage as shown in Figure 3.7.

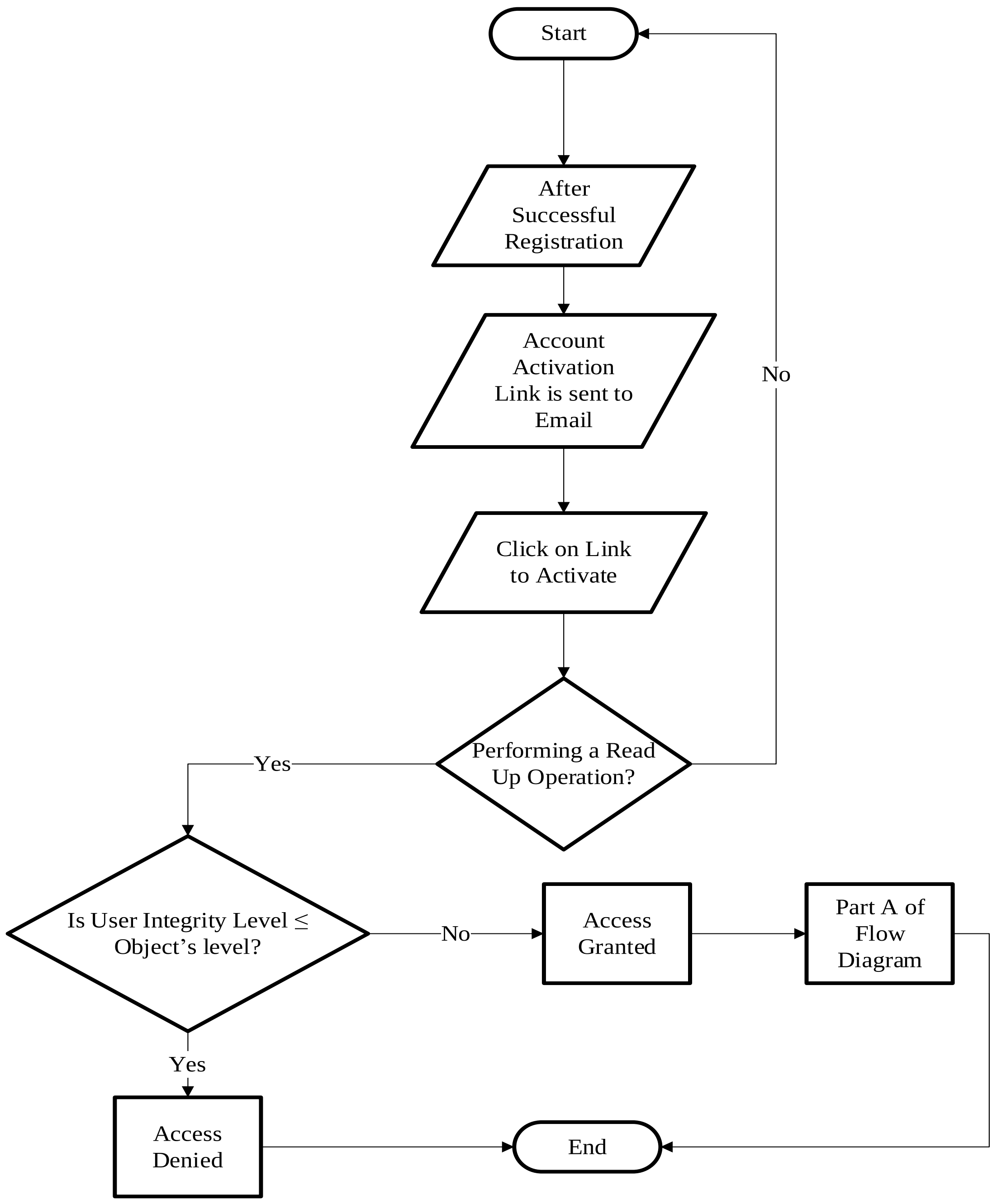

Start
After Successful Registration
Account Activation Link is sent to Email
Click on Link to Activate
Performing a Read Up Operation?
No
Yes
Is User Integrity Level ≤ Object's level?
No
Access Granted
Part A of Flow Diagram
Yes
Access Denied
End

Figure 3.7 Flow Diagram of the System Authentication

After a user has successfully registered into the system, an account activation link is sent to the user's email for activation. The user, therefore, has to open the email to activate the account. It moves to the decision-making process where the system asks whether the user wants to perform a read-up operation, if yes, the user's and object's integrity levels are checked, else, it goes back to start the process. Upon performing the integrity check, if the user's integrity level is less than or equal to the object's integrity level, the user is denied access to the system; else, the user is granted access to the software quality assurance model.

CHAPTER 4

SYSTEM IMPLEMENTATION, RESULTS, AND DISCUSSIONS

4.1 Introduction

This chapter deals with the system implementation, results, and discussions. The first aspect deals with ranking of the software quality attributes using the AHP. The second aspect deals with the implementation of each of the eleven (11) software quality attributes. Thirdly, the voting method is applied to combine the eleven (11) individually implemented quality attributes and are multiplied by criteria weights calculated using the AHP approach. Fourthly, the access control model is implemented to secure the overall system. Lastly, the performance of the secured quality assurance model is evaluated and validated using some standard metrics.

4.2 Identification of Software Quality Attributes

Twenty-four (24) software quality attributes were initially sampled using purposive sampling from ten (10) standard and well-known software quality models including FURPS model, Boehm's model, ISO-9126 quality model, Ghezzi's model, McCall's model, Dromey's model, Glib's model, Kitchenham and Pickard's model, Georgiadou's model, and Jamwal's model. This is shown in Table 4.1.

Table 4.1 Quality Attributes of the Existing Quality Models and the Proposed Model

Quality Models / Quality Attributes	McCall *et al.* (1977)	Boehm (1978)	FURPS (1987)	Dromey (1995)	ISO-9126 (1986)	Glib (1988)	Kitchenham and Pickard (1989)	Ghezzi, Jazayeri, and Mandrioli (1991)	Georgiadou (2003)	Jamwal and Jamwal (2009)	Proposed Hybrid Quality Model
Maintainability	/	/		/	/		/	/	/		/
Flexibility	/	/						/			*
Testability	/	/									/
Correctness	/									/	*
Reliability	/	/	/	/	/		/	/	/	/	/
Efficiency	/	/		/	/			/	/		/
Usability	/	/	/	/	/	/	/	/	/	/	/
Portability	/	/		/	/			/	/	/	/
Reusability	/			/				/			/
Interoperability	/										*
Understandability		/									*
Functionality			/	/	/				/		/
Performance			/							/	*
Supportability			/								*
Availability						/					/
Adaptability						/					*
Accuracy								/			*
Robustness									/	/	*
Extensibility			*								*
Security									/		/
Cost										/	/
Integrity	/							/			*
Confidentiality									/		*
Non-Repudiation									/		*

Table 4.1 shows the software quality attributes of the existing models and the proposed model. McCall's model addresses eleven (11) main attributes, Boehm's model has eight (8) main attributes, FURPS model has five (5) main attributes and one (1) sub-attribute, Dromey's model has seven (7) main attributes, the ISO 9126 model has six (6) main attributes, Glib's model has three (3) main attributes, Kitchenham and Pickard's model has three (3) main attributes, Ghezzi's model has nine (9) main attributes, Georgiadou's model has ten (10) main attributes, Jamwal's model has seven (7) main attributes while the proposed quality model has eleven (11) main attributes and thirteen (13) sub-attributes. Review of related works showed that some of the quality attributes had similar functionality as others and were grouped into main and sub-attributes. The following are the main quality attributes of the proposed model: Testability, Security, Efficiency, Reliability, Usability, Cost, Portability, Maintainability, Functionality, Reusability, and Availability. The sub-attributes are: Flexibility, Extensibility, Supportability, Integrity, Confidentiality, Non-Repudiation, Correctness, Interoperability, Robustness, Accuracy, Understandability, Performance and Adaptability.

4.3 Ranking of Software Quality Attributes Using Analytic Hierarchy Process

The judgement matrix was designed using thirty (30) experts' decisions, based on related research. The implementation was done in MATLAB/Simulink Software R2020b.

4.3.1 Quality Attribute Selection Judgement Matrices

A geometric mean of the scores from the survey was found and presented in a matrix form for effective criteria and pair-wise comparison and is represented in equation (4.1).

$$Q = \begin{bmatrix} 1.00 & 2.05 & 2.44 & 2.98 & 2.95 & 2.75 & 2.95 & 2.18 & 1.55 & 1.59 & 2.80 \\ 0.49 & 1.00 & 2.04 & 2.30 & 2.99 & 2.02 & 2.85 & 1.90 & 1.20 & 1.31 & 2.10 \\ 0.41 & 0.49 & 1.00 & 1.00 & 1.95 & 1.72 & 2.69 & 1.50 & 1.28 & 2.10 & 2.59 \\ 0.34 & 0.43 & 1.00 & 1.00 & 1.20 & 0.75 & 0.98 & 1.29 & 1.05 & 1.03 & 1.68 \\ 0.34 & 0.33 & 0.51 & 0.51 & 1.00 & 1.02 & 1.01 & 0.67 & 0.20 & 2.54 & 1.54 \\ 0.36 & 0.50 & 0.58 & 1.33 & 0.98 & 1.00 & 1.02 & 1.59 & 0.28 & 2.85 & 1.50 \\ 0.34 & 0.35 & 1.37 & 1.02 & 0.99 & 0.98 & 1.00 & 2.65 & 0.32 & 0.55 & 1.85 \\ 0.46 & 0.53 & 0.67 & 0.78 & 1.49 & 0.63 & 3.57 & 1.00 & 0.23 & 0.60 & 1.55 \\ 0.65 & 0.83 & 0.78 & 0.95 & 5.00 & 3.57 & 3.13 & 4.35 & 1.00 & 0.87 & 1.90 \\ 0.63 & 0.76 & 0.48 & 0.97 & 0.39 & 0.35 & 1.82 & 1.67 & 1.15 & 1.00 & 1.00 \\ 0.36 & 0.48 & 0.39 & 0.60 & 0.65 & 0.67 & 0.54 & 0.65 & 0.53 & 1.00 & 1.00 \end{bmatrix} \tag{4.1}$$

The geometric mean matrix is shown in Table 4.2. Table 4.3 also shows the normalised pair-wise comparison matrix while Table 4.4 shows the consistency matrix.

A questionnaire was administered to thirty (30) experts for the multi-criteria decision process. These experts filled the questionnaire by asking the importance of the quality attributes relative to the other but did not suggest that addition of other attributes to the model. The geometric mean of the scores from the filled questionnaire was found by multiplying the values for each of the attributes in Table 4.2 and setting it to the 1/nth power. The sum of each attribute was finally calculated. The geometric mean of the scores was found using equation (4.2).

$$\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}} = \sqrt[n]{x_1 x_2 \cdots x_n} \tag{4.2}$$

where, n = number of terms that are being multiplied; and

x = scores from the questionnaire.

Table 4.2 Geometric Mean of the Filled Questionnaire

Attributes	**M**	**T**	**R**	**P**	**A**	**E**	**Fn**	**Re**	**S**	**U**	**Co**
M	1.00	2.05	2.44	2.98	2.95	2.75	2.95	2.18	1.55	1.59	2.80
T	0.49	1.00	2.04	2.30	2.99	2.02	2.85	1.90	1.20	1.31	2.10
R	0.41	0.49	1.00	1.00	1.95	1.72	2.69	1.50	1.28	2.10	2.59
P	0.34	0.43	1.00	1.00	1.20	0.75	0.98	1.29	1.05	1.03	1.68
A	0.34	0.33	0.51	0.51	1.00	1.02	1.01	0.67	0.20	2.54	1.54
E	0.36	0.50	0.58	1.33	0.98	1.00	1.02	1.59	0.28	2.85	1.50
Fn	0.34	0.35	0.37	1.02	0.99	0.98	1.00	2.65	0.32	0.55	1.85
Re	0.46	0.53	0.67	0.78	1.49	0.63	3.57	1.00	0.23	0.60	1.55
S	0.65	0.83	0.78	0.95	5.00	3.57	3.13	4.35	1.00	0.87	1.90
U	0.63	0.76	0.48	0.97	0.39	0.35	1.82	1.67	1.15	1.00	1.00
Co	0.36	0.48	0.39	0.60	0.65	0.67	0.54	0.65	0.53	1.00	1.00
SUM	**5.36**	**7.75**	**10.26**	**13.44**	**19.60**	**15.46**	**21.56**	**19.44**	**8.79**	**15.44**	**19.51**

Table 4.2 shows the geometric mean of the scores from the questionnaire. Rules from the AHP show that the values on the diagonal of the table are always 1.00 while judgement values on the upper diagonal are the actual scores from the questionnaire. In order to fill the lower diagonal of the table, the reciprocal values of the upper diagonal are used. The sum of each column is also found and presented in the table. Availability was seen to have the highest number of 19.60 while Maintainability had the lowest number of 5.36.

Table 4.3 Normalised Pair-wise Comparison Matrix

Attributes	**M**	**T**	**R**	**P**	**A**	**E**	**Fn**	**Re**	**S**	**U**	**Co**	**Criteria Weight**	**Criteria Weight (%)**
M	0.186	0.26	0.238	0.222	0.151	0.178	0.1369	0.112	0.176	0.103	0.144	0.1737	**17.37**
T	0.091	0.13	0.199	0.171	0.153	0.131	0.1322	0.098	0.137	0.085	0.108	0.1302	**13.02**
R	0.076	0.06	0.098	0.074	0.100	0.111	0.1248	0.077	0.146	0.136	0.133	0.1035	**10.35**
P	0.063	0.06	0.098	0.074	0.061	0.049	0.0455	0.066	0.120	0.067	0.086	0.0713	**7.13**
A	0.063	0.04	0.05	0.038	0.051	0.066	0.0469	0.034	0.023	0.165	0.079	0.0599	**5.99**
E	0.068	0.06	0.057	0.099	0.050	0.065	0.0473	0.082	0.032	0.185	0.077	0.0749	**7.497**
Fn	0.063	0.05	0.036	0.076	0.051	0.063	0.0464	0.136	0.036	0.036	0.095	0.0622	**6.22**
Re	0.086	0.07	0.065	0.058	0.076	0.041	0.1657	0.051	0.026	0.039	0.079	0.0686	**6.86**
S	0.12	0.11	0.076	0.071	0.255	0.231	0.1450	0.224	0.114	0.056	0.097	0.1361	**13.61**
U	0.117	0.10	0.046	0.072	0.020	0.023	0.0844	0.086	0.131	0.065	0.051	0.0721	**7.22**
Co	0.067	0.06	0.038	0.044	0.033	0.043	0.0251	0.033	0.060	0.065	0.051	0.0473	**4.73**

The normalised pair-wise comparison matrix was found in Table 4.3 by diving each of the values for the attributes in Table 4.2 by the sum. To calculate the criteria weight, an average of the rows was found. The results indicated that, Maintainability had a criteria weight of 17.37% and Usability also had a criteria weight of 7.22%. Reusability, Availability, Testability and Functionality had 6.86%, 5.99%, 13.02% and 6.22% as their respective criteria weights. Also, Cost was seen to have 4.73%, Security also had 13.61% while Portability had 7.13%. Furthermore, Reliability and Efficiency were seen to have 10.35% and 7.49% respectively.

To evaluate the correctness of expert's evaluation, the consistency of the pair-wise comparison matrix was calculated in Table 4.4 by multiplying the criteria weight by the pair-wise comparison matrix, which was not normalised in Table 4.2. The weighted sum of the new matrix was found and then divided by the criteria weight in Table 4.3. The overall sum was found for the calculation of the consistency vector, λ_{max}, and Consistency Ratio (CR). The consistency vector, λ_{max}, was calculated by multiplying the pair-wise matrix by the weight vector and the sum of the row entries was divided by the corresponding criterion weight. The value of the consistency ratio must be less than 0.1 to make the judgement matrix acceptable.

$$\lambda_{\max} = \frac{134.04}{11} = 12.186 \tag{4.3}$$

$$\mathrm{CI} = \frac{\lambda_{\max} - \mathrm{n}}{\mathrm{n} - 1} = \frac{12.186 - 11}{10} = 0.119 \tag{4.4}$$

$$\mathrm{CR} = \frac{\mathrm{CI}}{\mathrm{RI}} = \frac{0.119}{1.52} = 0.079 \tag{4.5}$$

The selection judgement matrix is consistent since the value of the CR is 0.079, which is less than 0.1.

Results from Table 4.3 indicate that the attribute with the highest criteria weight of 17.37% is Maintainability (M) while the attribute with the lowest weight of 4.73% is Cost (C_o).

Table 4.4 Consistency of Pair-wise Comparison Matrix

Criteria Weight (CW)	**0.174**	**0.13**	**0.104**	**0.071**	**0.06**	**0.075**	**0.0622**	**0.069**	**0.136**	**0.072**	**0.047**		
Attributes	**M**	**T**	**R**	**P**	**A**	**E**	**Fn**	**Re**	**S**	**U**	**Co**	**Weighted Sum Value (WSV)**	**WSV/ CW**
M	0.174	0.27	0.253	0.213	0.177	0.206	0.1835	0.15	0.211	0.115	0.132	2.07981	11.973
T	0.085	0.13	0.211	0.164	0.179	0.151	0.1773	0.13	0.163	0.095	0.099	1.58551	12.178
R	0.071	0.06	0.104	0.071	0.117	0.129	0.1673	0.103	0.174	0.152	0.123	1.27414	12.308
P	0.058	0.06	0.104	0.071	0.072	0.056	0.0609	0.088	0.143	0.074	0.079	0.86401	12.116
A	0.059	0.04	0.053	0.037	0.06	0.076	0.0628	0.046	0.027	0.183	0.073	0.72068	12.029
E	0.063	0.06	0.06	0.095	0.059	0.075	0.0634	0.109	0.038	0.206	0.071	0.90389	12.057
Fn	0.059	0.05	0.038	0.073	0.059	0.074	0.0622	0.182	0.044	0.04	0.088	0.76337	12.274
Re	0.08	0.07	0.069	0.055	0.089	0.047	0.2221	0.069	0.031	0.043	0.073	0.84772	12.359
S	0.112	0.11	0.081	0.068	0.3	0.268	0.1944	0.298	0.136	0.063	0.09	1.71803	12.623
U	0.109	0.1	0.049	0.069	0.024	0.026	0.1131	0.114	0.156	0.072	0.047	0.88039	12.196
Co	0.062	0.06	0.04	0.042	0.039	0.05	0.0336	0.044	0.072	0.072	0.047	0.56433	11.929
SUM													**134.04**
$\lambda_{\max}$ **= 12.186**										**CR = 0.079**			

Figure 4.1 shows a graphical representation of the weights of the software quality attributes.

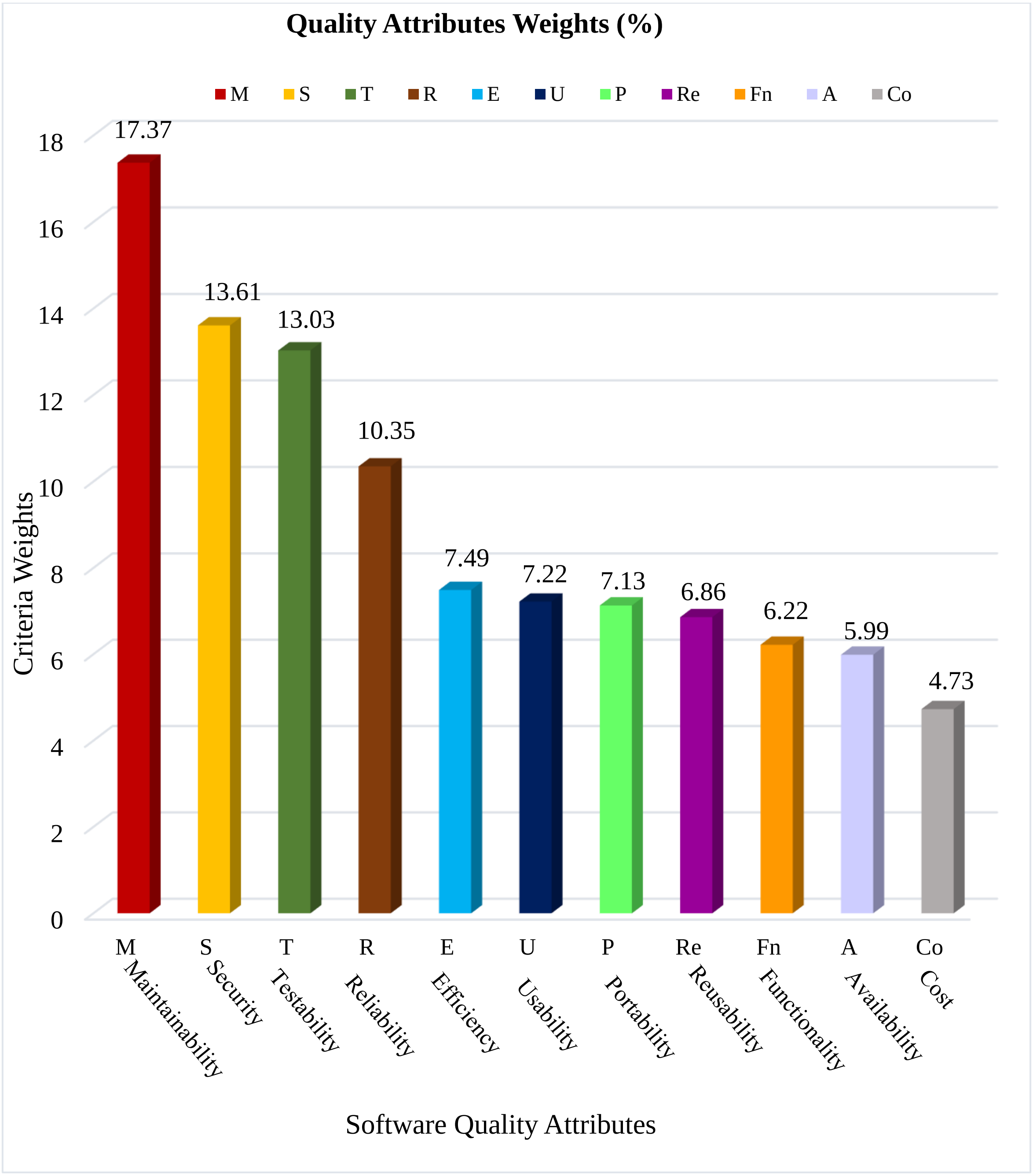


Figure 4.1 Weights of the Software Quality Attributes.

It can be seen from Figure 4.1 that Maintainability has the highest criteria weight of 17.37%. This is followed by Security with a criteria weight of 13.61%. Cost is noted to have the lowest weight of 4.73%. Quality attributes such as Testability, Reliability, Efficiency, Usability and Portability have percentage weights of 13.02, 10.35, 7.49, 7.22 and 7.13 respectively. Furthermore, Reusability and Functionality are noted to have criteria weights of 6.86% and 6.22%. Lastly, Availability had the second to last score with the weight 5.99%. According to experts judgement, Maintainability had the highest score while Cost had the lowest score. This was due to the fact that the users in this part of Africa (Ghana and Nigeria) are much concerned with using software with highly maintainable features but are less concerned with the cost of software because they mostly prefer the use of unlicensed software to licensed software. These results are consistent with the findings of Kassie and Singh (2020).

4.3.2 Alternative Selection Judgement Matrices

The alternatives, which are "Has Sub-attributes", "Doubles up as Sub-attributes", and "Mostly addressed", were also analysed for Maintainability as shown in Table 4.5.

Table 4.5 The Weight of Alternatives for Maintainability

Maintainability	**Has Sub-attributes**	**Doubles up as Sub-attributes**	**Mostly Addressed**	**Criteria Weight**
Has Sub-attributes	1	1	1/4	0.15
Doubles up as Sub-attributes	1	1	1/9	0.11
Mostly Addressed	4	9	1	0.74
$\lambda_{\max}$ **= 3.0749**		**CR = 0.0646**		

Table 4.5 indicates that Maintainability has been mostly addressed 74% of the time and has doubled up as a sub-attribute 11% of the time.

The alternatives were also analysed for Testability (T) as shown in Table 4.6.

Table 4.6 The Weight of Alternatives for Testability

Testability	**Has Sub-attributes**	**Doubles up as Sub-attributes**	**Mostly Addressed**	**Criteria Weight**
Has Sub-attributes	1	3	1/4	0.23
Doubles up as Sub-attributes	1/3	1	1/5	0.10
Mostly Addressed	4	5	1	0.67
$\lambda_{\max}$ = 3.0869	**CR = 0.07496**			

Table 4.6 shows that Testability has been mostly addressed at a rate of 67% and has doubled up as a sub-attribute at a rate of 10%.

The alternatives were also analysed for Reliability as shown in Table 4.7.

Table 4.7 The Weight of Alternatives for Reliability

Reliability	**Mostly Addressed**	**Has Sub-attributes**	**Doubles up as Sub-attributes**	**Criteria Weight**
Mostly Addressed	1	7	8	0.78
Has Sub-attributes	1/7	1	2	0.14
Doubles up as Sub-attributes	1/8	1/2	1	0.08
$\lambda_{\max}$ = 3.035	**CR = 0.0304**			

Table 4.7 shows that Reliability has been mostly addressed 78% times and has doubled up as a sub-attribute 8% times.

The alternatives were also analysed for Efficiency as shown in Table 4.8.

Table 4.8 The Weight of Alternatives for Efficiency

Efficiency	**Has Sub-attributes**	**Doubles up as Sub-attributes**	**Mostly Addressed**	**Criteria Weight**
Has Sub-attributes	1	4	1/3	0.28
Doubles up as Sub-attributes	1/4	1	1/5	0.10
Mostly Addressed	3	5	1	0.62
$\lambda_{\max}$ **= 3.0867** **CR = 0.0747**				

Table 4.8 shows that Efficiency is being mostly addressed 62% of the time and has doubled up as a sub-attribute 10% of the time.

The alternatives were also analysed for Usability as shown in Table 4.9.

Table 4.9 The Weight of Alternatives for Usability

Usability	**Mostly Addressed**	**Has Sub-attributes**	**Doubles up as Sub-attributes**	**Criteria Weight**
Mostly Addressed	1	9	8	0.80
Has Sub-attributes	1/9	1	2	0.12
Doubles up as Sub-attributes	1/8	1/2	1	0.08
$\lambda_{\max}$ **= 3.075** **CR = 0.0649**				

Table 4.9 shows that Usability has been mostly addressed at a rate of 80% and has doubled up as a sub-attribute at a rate of 8%.

The alternatives were also analysed for Portability as shown in Table 4.10.

Table 4.10 The Weight of Alternatives for Portability

Portability	**Has Sub-attributes**	**Doubles up as Sub-attributes**	**Mostly Addressed**	**Criteria Weight**
Has Sub-attributes	1	5	1/2	0.35
Doubles up as Sub-attributes	1/5	1	1/5	0.093
Mostly Addressed	2	5	1	0.56
$\lambda_{\max}$ **= 3.0183**	**CR = 0.01581**			

Table 4.10 shows that Portability is being mostly addressed at a rate of 56% and has doubled up as a sub-attribute at a rate of 9%.

The alternatives were also analysed for Reusability as shown in Table 4.11.

Table 4.11 The Weight of Alternatives for Reusability

Reusability	**Mostly Addressed**	**Doubles up as Sub-attributes**	**Has Sub-attributes**	**Criteria Weight**
Mostly Addressed	1	1	1	0.33
Doubles up as Sub-attributes	1	1	1/2	0.26
Has Sub-attributes	1	2	1	0.41
$\lambda_{\max}$ **= 3.054**	**CR = 0.0463**			

Table 4.11 shows that Reusability has sub-attributes at a rate of 41% and has doubled up as a sub-attribute at a rate of 26%.

The alternatives were also analysed for Functionality as shown in Table 4.12.

Table 4.12 The Weight of Alternatives for Functionality

Functionality	**Has Sub-attributes**	**Doubles up as Sub-attributes**	**Mostly Addressed**	**Criteria Weight**
Has Sub-attributes	1	4	1/2	0.33
Doubles up as Sub-attributes	1/4	1	1/5	0.10
Mostly Addressed	2	5	1	0.57
$\lambda_{\max}$ = 3.0247 CR = 0.0213				

Table 4.12 shows that Functionality has been mostly addressed at a rate of 57% and has doubled up as a sub-attribute at a rate of 10%.

The alternatives were also analysed for Availability as shown in Table 4.13.

Table 4.13 The Weight of Alternatives for Availability

Availability	**Mostly Addressed**	**Has Sub-attributes**	**Doubles up as Sub-attributes**	**Criteria Weight**
Mostly Addressed	1	1	1/5	0.17
Has Sub-attributes	1	1	1/2	0.23
Doubles up as Sub-attributes	5	2	1	0.60
$\lambda_{\max}$ = 3.0951 CR = 0.08196				

Table 4.13 shows that Availability has doubled up as a sub-attribute at a rate of 60% and has been mostly addressed at a rate of 17%.

The alternatives were also analysed for Cost as shown in Table 4.14.

Table 4.14 The Weight of Alternatives for Cost

Cost	**Has Sub-attributes**	**Mostly Addressed**	**Doubles up as Sub-attributes**	**Criteria Weight**
Has Sub-attributes	1	1	1/9	0.11
Mostly Addressed	1	1	1/5	0.13
Doubles up as Sub-attributes	9	5	1	0.77
$\lambda_{\max}$ **= 3.0389**	**CR = 0.0336**			

Table 4.14 shows that Cost has doubled up as a sub-attribute at a rate of 77% and has sub-attributes at a rate of 11%.

The alternatives were also analysed for Security as shown in Table 4.15.

Table 4.15 The Weight of Alternatives for Security

Security	**Mostly Addressed**	**Has Sub-attributes**	**Doubles up as Sub-attributes**	**Criteria Weight**
Mostly Addressed	1	2	1/2	0.35
Has Sub-attributes	1/2	1	1/9	0.09
Doubles up as Sub-attributes	2	9	1	0.56
$\lambda_{\max}$ **= 3.0745**	**CR = 0.0642**			

Table 4.15 shows that Security has doubled up as a sub-attribute at a rate of 56% and has sub-attributes at a rate of 9%.

The overall weights for the software quality attribute selection are summarised in Table 4.16.

Table 4.16 The Weights for Software Quality Attribute Selection

Element	**Weight**
Alternatives	
Mostly Addressed	0.5200
Doubles up as Sub- attributes	0.2210
Has Sub-attributes	0.2590
Criteria or Attributes	
Maintainability	0.1737
Testability	0.1302
Reliability	0.1035
Efficiency	0.0749
Usability	0.0722
Portability	0.0713
Reusability	0.0686
Security	0.1361
Functionality	0.0622
Availability	0.0599
Cost	0.0473
Overall Consistency Ratio: 0.0570	

The overall consistency ratio, $\overline{CR}$, was calculated as shown in equation (4.6) by summing up the weighted consistency index, w_iCI_i, in the nominator and the weighted random consistency index, w_iRI_i, in the denominator.

$$\overline{CR} = \frac{\sum_i w_i CI_i}{\sum_i w_i RI_i} \tag{4.6}$$

The results in Table 4.16 show that "Mostly Addressed" is the highest-ranking software quality alternative with 0.520 representing 52% and "Has Sub- attribute" is the lowest ranking alternative with 0.221 representing 22.10%. The result also shows Maintainability as the highest-ranking software quality attribute with 0.1737 which represents 17.37%. Table 4.16 also shows that the overall analysis is consistent since the value of CR is 0.057, which is less than 0.1. The analysis can, therefore, be considered as consistent.

4.3.3 Quality models and attributes

Software quality attributes such as Usability, Maintainability, and Reliability have been addressed by most quality models while Availability, Security, and Cost have been addressed by single quality models. This is shown in Figure 4.2.

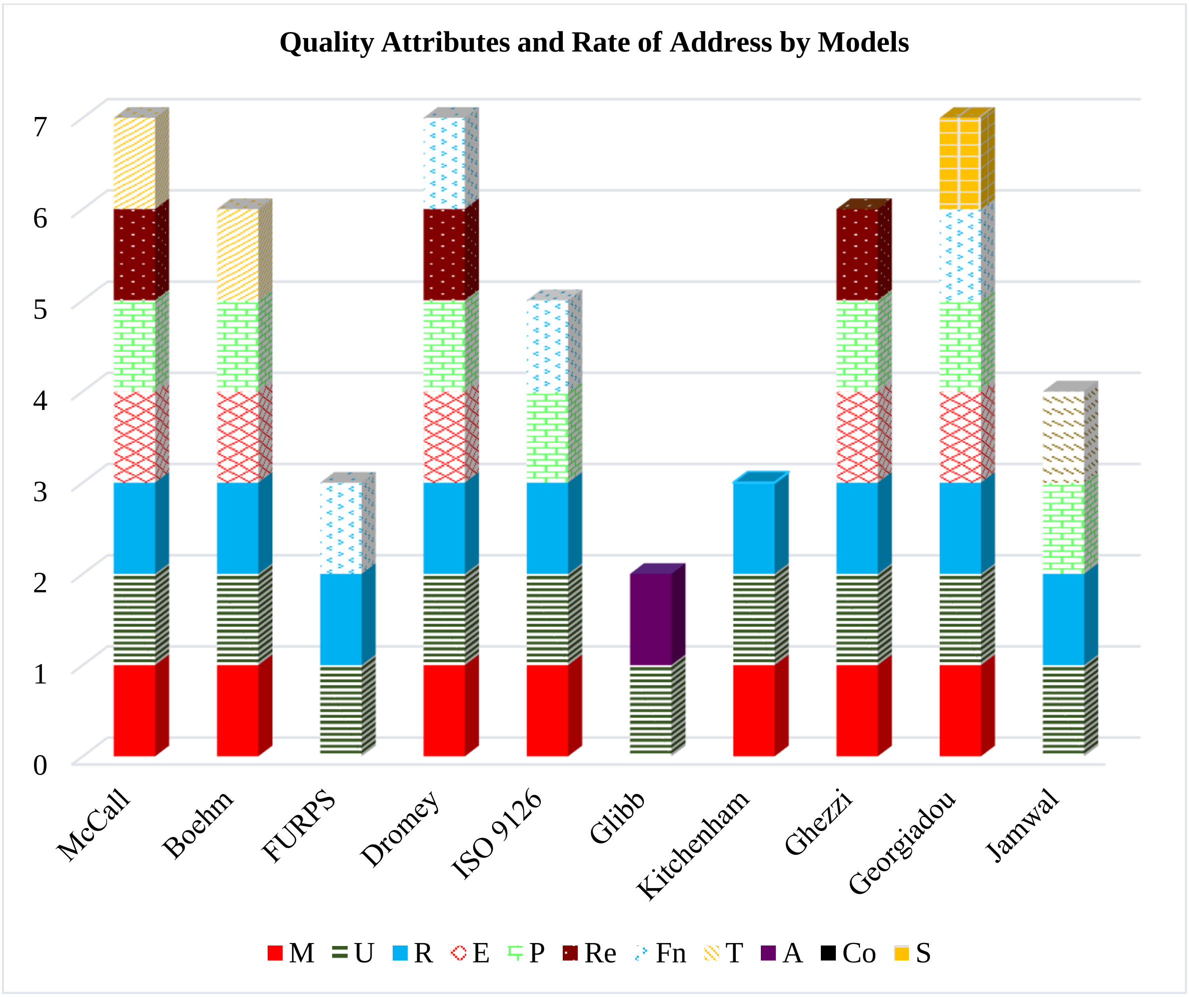


Figure 4.2 Quality attributes and their rates of address by Quality models

Usability has been addressed by all the quality models, followed by Reliability which has also been addressed by nine (9) out of ten (10) models. Figure 4.2 also shows that Availability has only been addressed by Glib's model, Security has been addressed by Georgiadou while Cost has also been only addressed by Jamwal's model.

Maintainability has been mostly addressed in software quality models such as McCall’s model, Boehm’s model, Dromey’s model, ISO-9126 model, Kitchenham and Pickard’s model, Ghezzi’s model, and Georgiadou’s model as shown in Figure 4.2.

Usability has been addressed in all the software quality models as shown in Figure 4.2. It has always been present, even in the very first software quality models, and is also one of the widely used software quality attributes in the industry.

According to Figure 4.2, Reliability has also been addressed by most software quality models such as McCall’s model, Boehm’s model, Dromey’s model, FURPS, ISO-9126 model, Kitchenham and Pickard’s model, Ghezzi’s model, Georgiadou’s model, and Jamwal’s model.

Efficiency has been addressed by McCall’s model, Boehm’s model, Dromey’s model, Ghezzi’s model, and Georgiadou’s model.

Portability has been addressed by McCall’s model, Boehm’s model, Dromey’s model, ISO-9126 model, Ghezzi’s model, Georgiadou’s model, and Jamwal’s model.

Reusability, according to Figure 4.2, has been addressed by McCall’s model, Dromey’s model, and Ghezzi’s model.

Functionality has also been addressed by the FURPS model, Dromey’s model, ISO-9126 model, and Georgiadou’s model as shown in Figure 4.2.

Testability has been addressed by McCall’s model and Boehm’s model.

Availability has been addressed by Glib’s model only.

Security has been addressed by Georgiadou’s model. It is seen as an important software quality attribute due to its enforcement of confidentiality, integrity, authentication, and non-repudiation schemes into software products.

Cost has also been addressed by Jamwal’s model only.

4.4 Software Quality Assurance Model Implementation

The implementation of the software quality model was done using Python and the interface was designed using web technologies such as ExpressJS, Angular, and NodeJS. Data was stored using MongoDB. The application can run on browsers such as Mozilla Firefox, Google Chrome, Microsoft Edge, Internet Explorer, Safari, and Opera Mini. The following are the other requirements for running the software:

a) Operating System Requirements: It can run on Windows 7, Windows 8, or Windows 10 and Mac OSX 10.8, 10.9, 10.10 or 10.11.
b) Hardware Requirements: It can be run on a laptop with a processor speed of 2.3 Gigahertz (GHz) or above, a minimum of 2 GB RAM, monitor resolution of 1024×768 or higher, Ethernet connection (LAN) or wireless adapter (WiFi) with a speed of 4 Mbps or higher.

The user login page contains text fields for users to enter their email and password as shown in Figure 4.3 before having access to the quality assurance software. It has been secured using a hybrid security model consisting of Bell-LaPadula and Biba models. There are also options for registering new users and an option for resetting one's password in case of forgetting the password.

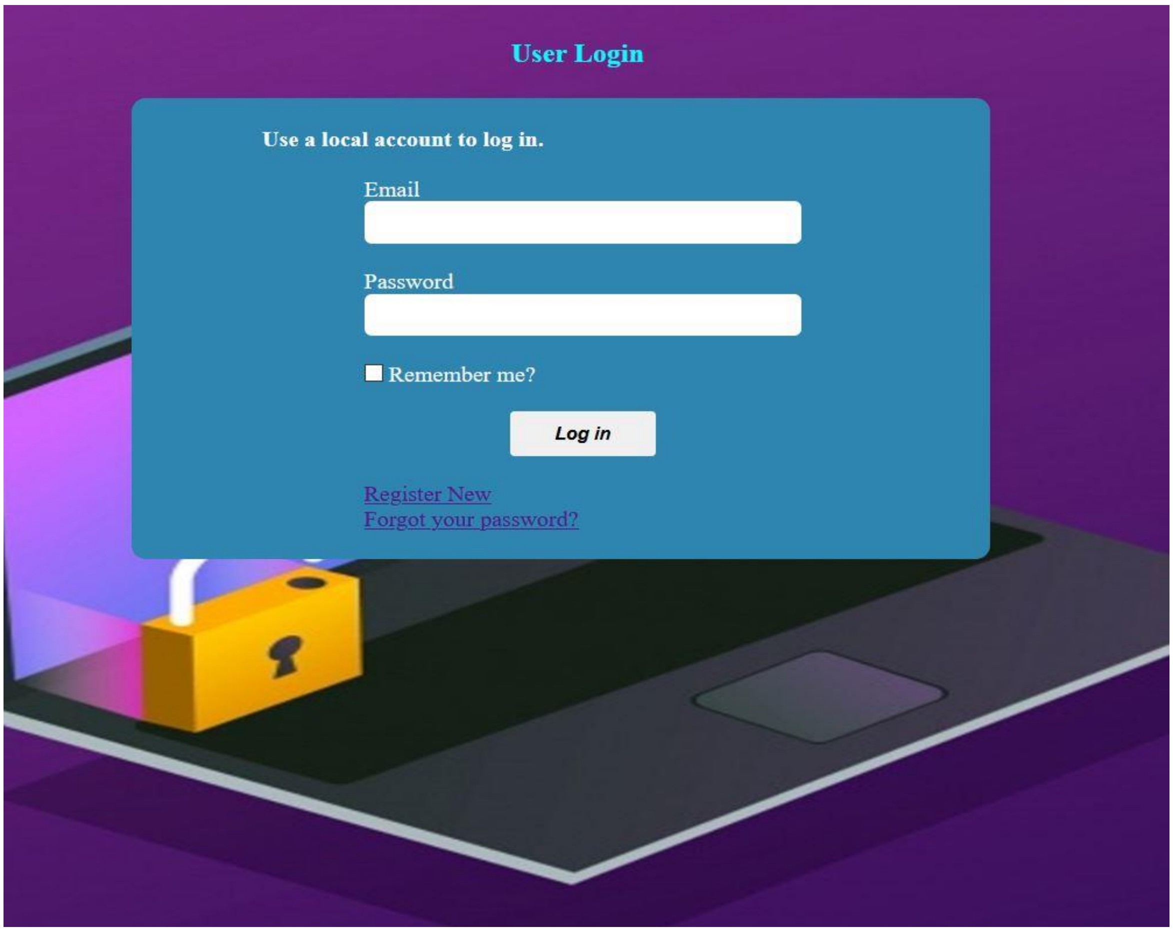


Figure 4.3 User Login Page of the Software Quality Assurance Model

The register account page as shown in Figure 4.4 allows new users to register into the quality assurance software. It contains text fields for entering one's username, email address, password, and confirmation email. Once the register button is clicked, the user is notified as having registered successfully as shown in Figure 4.5.

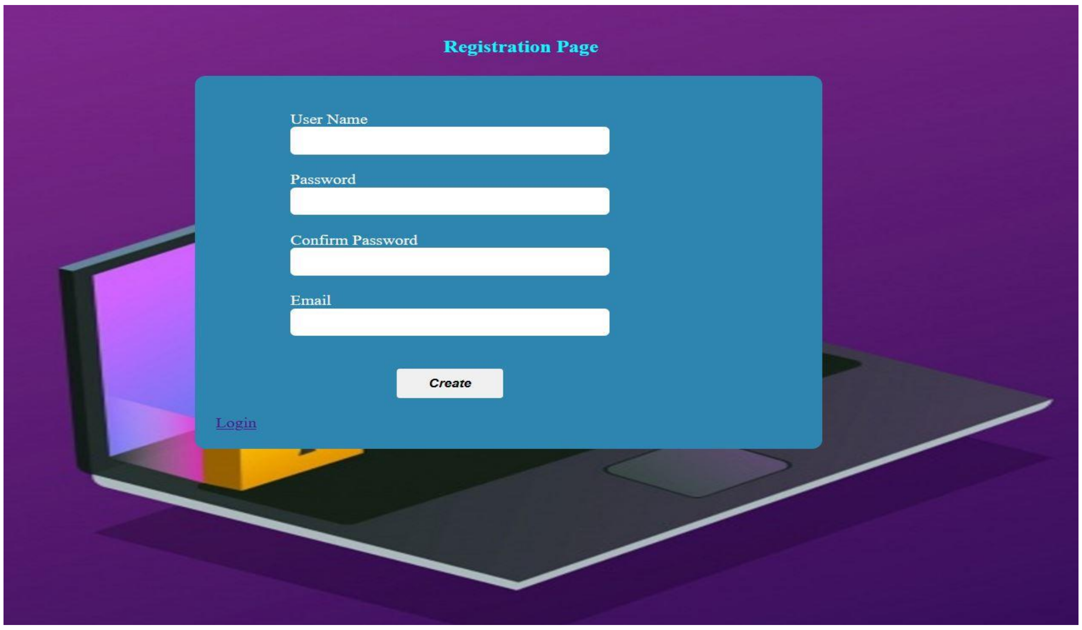
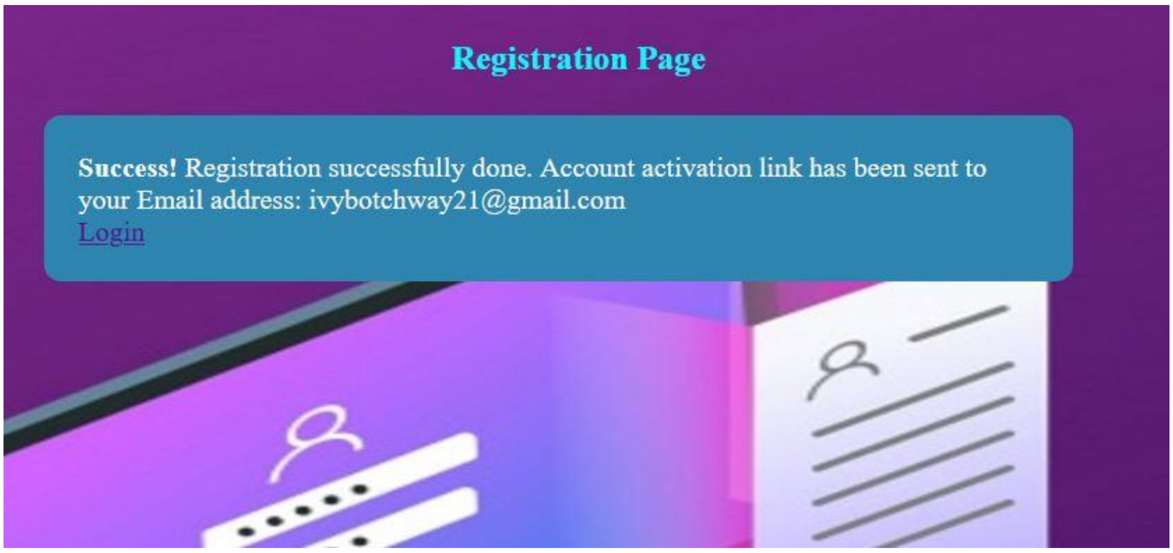


Figure 4.4 Register Account Page of the Software Quality Assurance Model

Upon successful login, an account activation link is sent to the user's email address for authentication. This is shown in Figure 4.5.


Registration Page

Success! Registration successfully done. Account activation link has been sent to your Email address: ivybotchway21@gmail.com
Login

Figure 4.5 Successful Register Account Page

4.4.1 Security Model for Quality Assurance Software Access Control

Authentication of user account registration is done through two-factor authentication. The first factor is the textual password entry while the second one is the account activation link sent to the user's email address.

Ensuring Confidentiality

The confidentiality of the quality assurance software was enforced using the BLP model to ensure that unauthorised persons are denied access privilege to the software. This allows an authorised person to log into the system and run the test for a web application's quality assurance assessment. The confidentiality check uses the following process:

a) A user sends a request to the permission granting engine, which contains the BLP write-up commands.
b) Upon receiving the request, the permission granting engine performs a check to match the user's confidentiality level to the object's confidentiality level.
c) Once the user's confidential security level is not less than the object's confidentiality level, access to the system is granted. If the user's confidential security level is less than or equal to the object's confidentiality level, access to the system is denied.

Ensuring Integrity

Integrity was also enforced based on the Biba model to prevent unauthorised persons from altering data in the quality assurance software. This allows an authorised person to receive an activation code for successful login to the software after completing the system registration process. It uses the following process:

a) An account activation code is sent to the user's email address.
b) The user opens the email and clicks on the code for activation.
c) The permission granting engine, which also contains the Biba read-up commands, performs a check to match the user's integrity level to the object's integrity level.
d) Once the user's integrity level is not less than the object's integrity level, read access to the system is granted. If the user's integrity security level is less than or equal to the object's integrity level, read access to the system is denied.

Confusion Matrix for the Access Control

A confusion matrix was generated using Scikit-learn in Python programming language and the result is shown in Table 4.17.

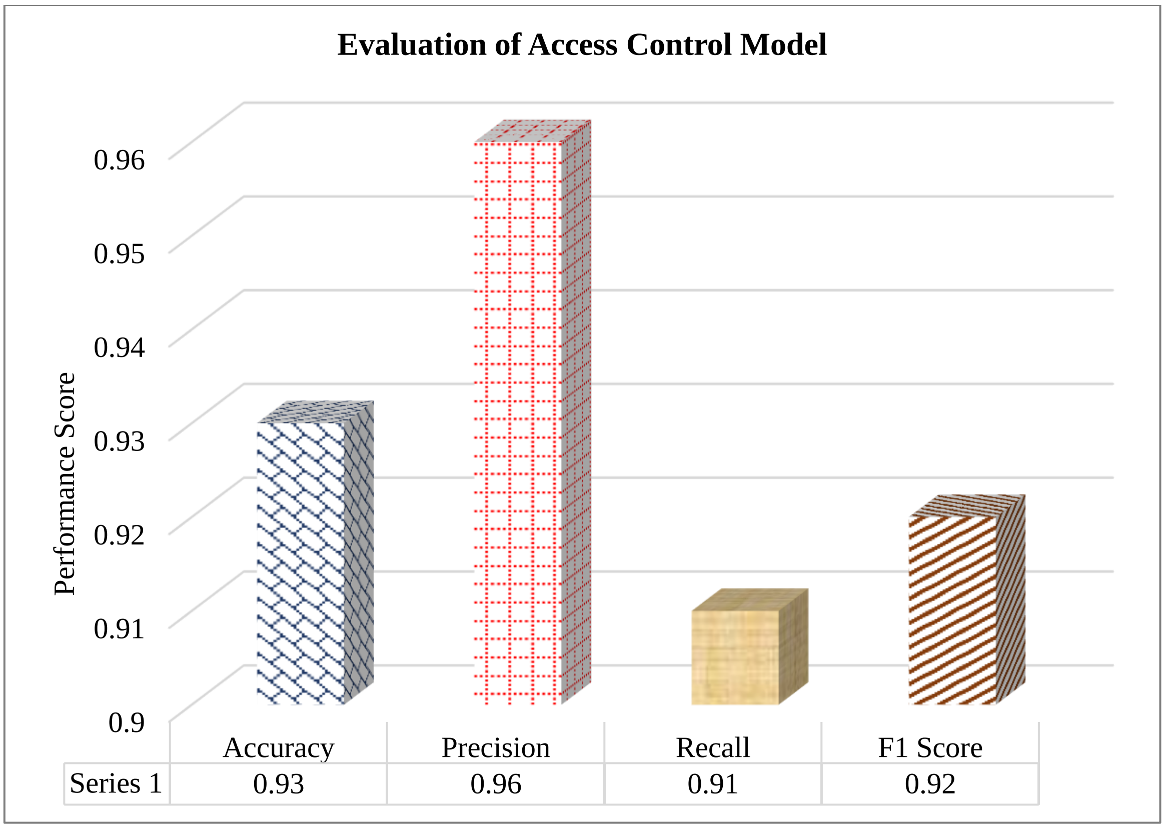
Table 4.17 Confusion Matrix for Access Control

Results	**Access Granted**	**Access Denied**
Access Granted	50	2
Access Denied	5	43

According to Table 4.17, the confusion matrix recorded the user data points for access denied and access granted values. 50 positive user data points were correctly granted access to the software (true positive); 43 negative user data points were correctly denied access to the software (true negative); 2 negative user data points as incorrectly granted access to the software while 5 positive user data points were incorrectly denied access to the software.

In reality, 100 predictions were made and out of it, the classifier predicted the denied access value to be 45 and predicted access granted 55 times. There were 52 cases in which the actual value was Access granted and 48 cases in which the actual value was Access denied.

In order to evaluate the performance of the access control, the accuracy, precision, recall and F1 score of the confusion matrix were calculated and shown in Figure 4.6:



	Accuracy	Precision	Recall	F1 Score
Series 1	0.93	0.96	0.91	0.92

Figure 4.6 Performance Evaluation of Access Control Model

Figure 4.6 shows a pictorial representation of evaluation of the access control model where the precision, accuracy, recall and f1 scores are shown. Accuracy had a score of 0.93, Precision had 0.96, Recall had 0.91 and F1 score had 0.92.

Quality Assurance Model Homepage

The homepage of the software quality assurance model after successful login is shown in Figure 4.7. It shows the identity of the user by displaying the user's email address in the upper right corner.

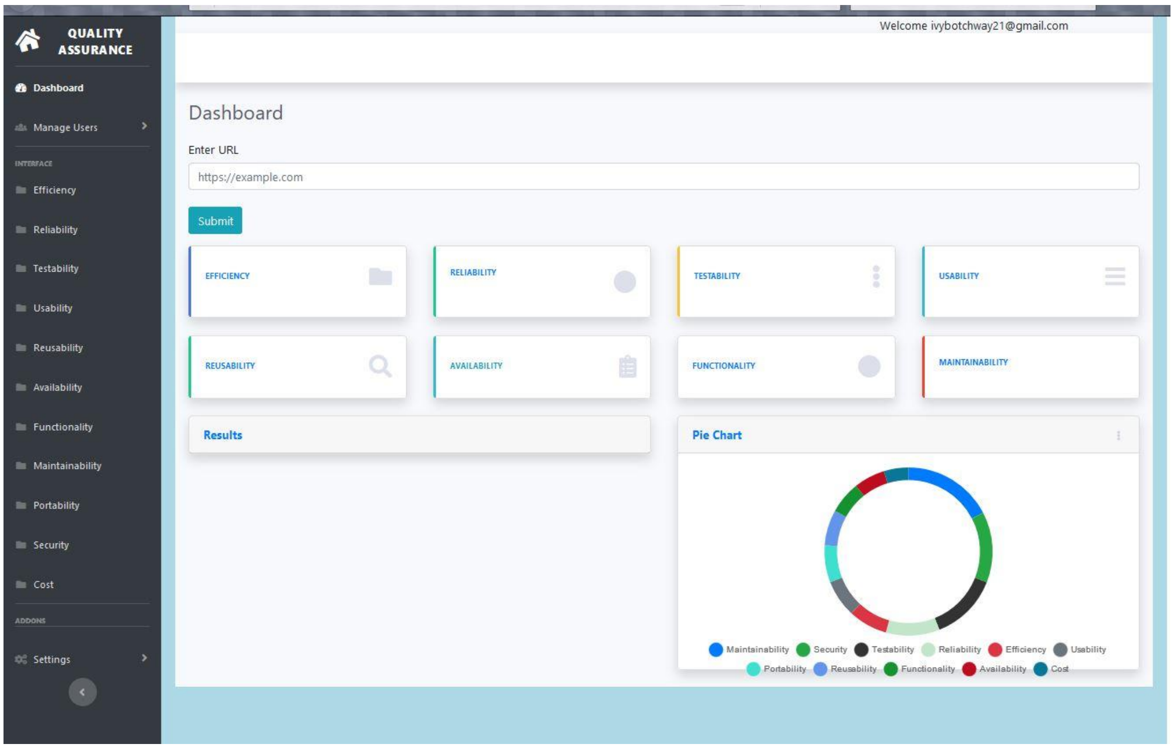
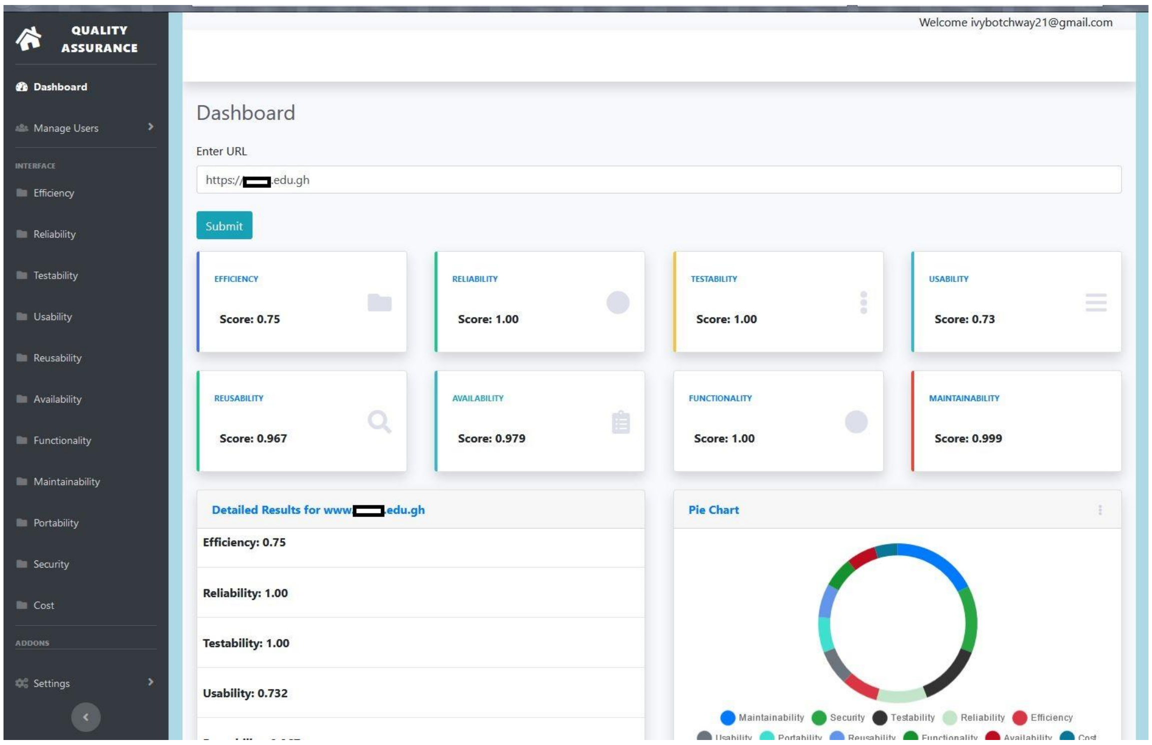


Figure 4.7 Homepage of Software Quality Assurance Model

The homepage contains three (3) important sections: Manage Users, Interface, Settings, and Detailed results.

a) Manage Users: This section allows the creation, editing, and deleting of user functions;
b) Interface: This section contains the eleven (11) software quality attributes. Efficiency, reliability, testability, usability, reusability, availability, functionality, maintainability, portability, security, and software cost assessment are performed in this section;
c) Settings: This section contains the login and logout functions; and
d) Detailed results: This is where the detailed results of the quality assessment are displayed.

The generated results are shown in Figure 4.8 and Figure 4.9.



Welcome ivybotchway21@gmail.com

Dashboard

Enter URL

https://[redacted].edu.gh

EFFICIENCY Score: 0.75

RELIABILITY Score: 1.00

TESTABILITY Score: 1.00

USABILITY Score: 0.73

REUSABILITY Score: 0.967

AVAILABILITY Score: 0.979

FUNCTIONALITY Score: 1.00

MAINTAINABILITY Score: 0.999

Detailed Results for www[redacted].edu.gh

Efficiency: 0.75

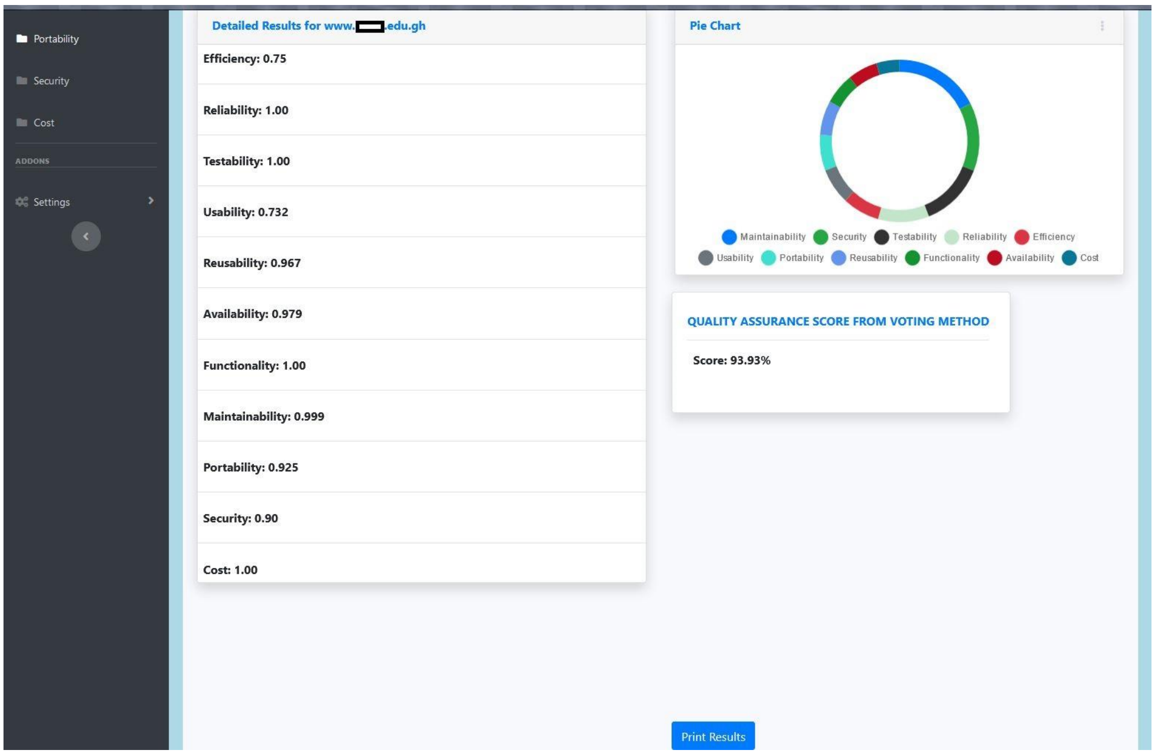
Reliability: 1.00

Testability: 1.00

Usability: 0.732

Figure 4.8 Generated Results showing Scores of Attributes for a Domain Name

Figure 4.8 pictorially represents the results generated for a domain name by displaying the scores against the software quality attribute’s name.



Detailed Results for www.[redacted].edu.gh

Efficiency: 0.75

Reliability: 1.00

Testability: 1.00

Usability: 0.732

Reusability: 0.967

Availability: 0.979

Functionality: 1.00

Maintainability: 0.999

Portability: 0.925

Security: 0.90

Cost: 1.00

QUALITY ASSURANCE SCORE FROM VOTING METHOD

Score: 93.93%

Figure 4.9 Results showing Score from Voting Model

According to Figure 4.9, the score generated from the voting model is displayed. The Figure also displays all scores generated for the eleven-software quality attributes and allows the user to perform a print function of the generated results.

Web-based Applications for Evaluation

The web applications being evaluated sum up to twenty-eight (28) and have been grouped under six (6) categories: Educational web applications, Video editing web applications, E-commerce software, Online form creation software, Company web applications, and Document creation software. The web applications and their corresponding masked domain names are shown in Table 4.18.

Table 4.18 List of Web Applications being evaluated

Name	**Domain Name**
Educational Web Applications	
Application 1	https://www. application1.edu.gh/
Application 2	https://www. application2.edu.ng/
Application 3	https://www. application3.edu.gh/
Application 4	https://www. application4.edu.gh/
Application 5	https://www.application5.edu.ng/
Video editing Web applications	
Application 6	https://www.application6.com/
Application 7	https://www.application7.com/
Application 8	https://www.application8.com/
Application 9	https://www.application9.com/
Application 10	https://www.application10.com/
E-commerce Software	
Application 11	https://www.application11.com.ng/
Application 12	https://www.application12.com/
Application 13	https://www.application13.com/
Application 14	https://www.application14.com/
Application 15	https://www.application15.com/

Table 4.18 List of Web Applications being evaluated (cont'd)

Name	**Domain Name**
Online Form Creation Software	
Application 16	https://www.application16.com/
Application 17	https://www.application17.com/
Application 18	https://www.application18.com/
Application 19	https://www.application19.com/
Application 20	https://www.application20.com/
Company Web applications	
Application 21	https://www.application21.com
Application 22	http://www.application22.com/
Application 23	https://www.application23.com/
Application 24	http://www.application24.com/
Application 25	http://www.application25.com/
Document Creation Software	
Application 26	https://application26.com/
Application 27	http://www.application27.com/
Application 28	https://www.application28.com/

4.4.2 Software Efficiency Test

The throughput and bandwidth for the web applications were evaluated and the overall Efficiency was calculated. The python script for efficiency evaluation in the custom application mirrors what users are expected to do, such as navigating through a web application, searching for an item, registering for an account among others. Upon entering the URL of a website into the application, the python script opens the website and performs some activities to mimic a typical user on a website.

These activities were then automated and run for 500 users and the number of requests being sent over time to the web server of the website under review was measured as the throughput. The characters sent per second to the web server was also recorded as the bandwidth.

The efficiency test performed is network dependent, hence, better results were recorded with good network while poor network gave poor results. Therefore, the test was performed on wireless (Wi-Fi) and cellular internet networks respectively and an average of the scores was found and is shown in Table 4.19.

Table 4.19 Web Applications and Overall Efficiency

Application Name	**Throughput (KiB/s)**	**Bandwidth (KiB/s)**	**Efficiency Score**	**Efficiency (%)**
Educational Web Applications				
Application 1	236.25	314.98	0.75	75
Application 2	178.56	278.63	0.64	64
Application 3	206.61	268.32	0.77	**77**
Application 4	165.59	217.88	0.76	76
Application 5	131.58	185.32	0.71	71
Video editing Web applications				
Application 6	136.37	197.63	0.69	69
Application 7	172.61	243.11	0.71	71
Application 8	156.72	195.89	0.80	80
Application 9	154.46	217.54	0.71	71
Application 10	137.11	214.23	0.64	64
E-commerce Software				
Application 11	224.17	339.65	0.66	66
Application 12	158.08	254.96	0.62	62
Application 13	177.642	236.856	0.75	75
Application 14	187.52	215.53	0.87	87
Application 15	161.67	218.47	0.74	74
Online Form Creation Software				
Application 16	177.35	218.95	0.81	81
Application 17	162.93	214.37	0.76	76
Application 18	148.79	198.38	0.75	75
Application 19	140.95	195.76	0.72	72
Application 20	153.77	216.57	0.71	71

Table 4.19 Web Applications and Overall Efficiency (cont'd)

Application Name	Throughput (KiB/s)	Bandwidth (KiB/s)	Efficiency Score	Efficiency (%)
Company Web applications				
Application 21	176.08	217.38	0.81	81
Application 22	149.63	204.96	0.73	73
Application 23	149.38	196.54	0.76	76
Application 24	188.05	213.69	0.88	88
Application 25	147.18	193.65	0.76	76
Document Creation Software				
Application 26	138.15	206.18	0.67	67
Application 27	146.49	160.97	0.91	91
Application 28	95.65	156.79	0.61	61

Table 4.19 shows that the Efficiency for each of the web applications has been evaluated. For the Educational web applications, Application 2 had an Efficiency score of 64% with 178.56KiB/s throughput and bandwidth of 278.63KiB/s. Application 1 had 75% as its Efficiency score with 236.25KiB/s throughput and bandwidth of 314.98KiB/s. The throughput score for Application 3 was 206.61KiB/s and bandwidth was 268.32KiB/s with an Efficiency of 77%. Application 4 had an Efficiency score of 76% with a throughput of 165.59KiB/s and bandwidth of 217.88KiB/s. Finally, Application 4 had an Efficiency score of 71% with a throughput of 131.58KiB/s and bandwidth of 185.32KiB/s. The application with the highest efficiency score was Application 3 while Application 2 had the lowest score. This shows that Application 3 can perform well and provide faster results while using less computing resources than the other applications. The average efficiency score in the category is 72.60%.

For the Video editing web applications, the Efficiency score of Application 6 was 0.69 with a throughput of 136.37KiB/s, a bandwidth of 197.63KiB/s, and an Efficiency of 69%. Application 7 had 71% as its Efficiency score, 172.61KiB/s as throughput, and bandwidth of 243.11KiB/s. Application 8 also had 80% as its Efficiency score with 156.72KiB/s as throughput and bandwidth of 195.89KiB/s. The throughput score for Application 9 was 154.46KiB/s and bandwidth was 217.54KiB/s. Finally, Application 10 had an Efficiency score of 64% with a throughput of 137.11KiB/s and bandwidth of 214.63KiB/s. It can be seen that Application 8 had the highest Efficiency score under the category while Application 10 had the lowest score. The average efficiency score in the category is 71%.

For the E-commerce software, The Efficiency level of Application 11 was 0.66 with a throughput of 224.17KiB/s, a bandwidth of 339.65KiB/s, and an Efficiency of 66%. Application 12 had 62% as its Efficiency score, 158.08KiB/s as throughput, and bandwidth of 254.96KiB/s. Application 13 also had 75% as its Efficiency score with 177.642KiB/s as throughput and bandwidth of 236.85KiB/s. The throughput score for Application 14 was 187.52KiB/s and bandwidth was 215.53KiB/s with an Efficiency level of 0.87. Finally, Application 15 had an Efficiency score of 74% with a throughput of 161.67KiB/s and bandwidth of 218.47KiB/s. Application 14 had the highest efficiency score with 87% while Application12 had the lowest score with 62%. The average Efficiency score is seen to be 72.80%.

The evaluated online form creation software had efficiency scores ranging between 70% and 82% with an average Efficiency score of 75%. The throughput also ranged between 140KiB/s and 177KiB/s while bandwidth ranged between 195KiB/s and 219KiB/s. Application 16 was seen to have a throughput of 177.35KiB/s and bandwidth of 218.95KiB/s and an efficiency score of 81%. Application 17 had a throughput of 162.93KiB/s and bandwidth of 214.37KiB/s and an efficiency score of 76%. Application 18 had a throughput of 148.79KiB/s and bandwidth of 198.38KiB/s and an efficiency score of 75%. Application 19 had a throughput of 140.95KiB/s and bandwidth of 195.76KiB/s and an efficiency score of 72%. Finally, the efficiency score of Application 20 was 71% with throughput and bandwidth of 153.77KiB/s and 216.57KiB/s respectively. Application 16 had the highest Efficiency score while Application 20 had the lowest score.

The company web applications had Efficiency levels ranging between 0.72 and 0.89. The highest throughput value was 176.08 KiB/s while the lowest was 147.18KiB/s. The highest bandwidth score was recorded as 217.38KiB/s and the lowest was 193.65KiB/s. Application 24 had the highest Efficiency score of 88% with throughput and bandwidth of 188.05KiB/s and 213.69KiB/s respectively while Application 22 had the lowest Efficiency score of 73% with throughput and bandwidth of 149.63KiB/s and 204.96KiB/s respectively. The average Efficiency score was 78.80%.

The Document Creation Software also performed the test with Efficiency levels ranging from 0.61 to 0.91. The highest efficiency score was recorded as 91% while the lowest was 61%. Application 27 was seen to have performed better among all the evaluated Document Creation Software with an Efficiency score of 91%, throughput of 146.49KiB/s, and bandwidth of 160.97KiB/s. Application 26 also had an Efficiency score of 67%, throughput of 138.15KiB/s, and bandwidth of 206.18KiB/s. Finally, Application 28 had the lowest Efficiency score of 61% with a throughput of 95.65KiB/s and bandwidth of 156.79KiB/s. The average Efficiency score was seen to be 73%.

The web application category with the highest average Efficiency score was the Company web application with 78.80% while the one with the lowest average Efficiency score was the Video editing web applications with 71%.

4.4.3 Software Reliability Test

The applications were evaluated for reliability by conducting a stress test to assess the ability of the website to cope well under stress. The stress test evaluates the error handling capabilities of the website under extremely heavy conditions (accessed by a lot of users concurrently) and ensures that the application does not crash under such instances.

The python script conducted the test by creating the test plan where the number of threads (number of users), the ramp up period (time it takes a thread to begin execution) and the loop count (how many times to repeat the test) were provided. Finally, an HTTP get request was sent to the website under review.

The test was run for 500 concurrent users and the number of users that failed the test was recorded as the failure rate. A failure rate of 0 shows that all the users could concurrently access the web application without failure. A failure rate of 120 showed that 120 out of 500 users could not access the web application and the system experienced failure. The reliability test done on each of the web applications is shown in Table 4.20.

Table 4.20 Reliability Test Evaluation

Application Name	**Testing Time, t (s)**	**Failure Rate (λ)**	**Score**	**Reliability Score (%)**
Educational Web Applications				
Application 1	120	0	1	100
Application 2	120	120	0	0
Application 3	120	130	0	0
Application 4	120	0	1	100
Application 5	120	0	1	100
Video editing Web applications				
Application 6	120	0	1	100
Application 7	120	0	1	100
Application 8	120	0	1	100
Application 9	120	0	1	100
Application 10	120	0	1	100
E-commerce Software				
Application 11	120	0	1	100
Application 12	120	70	0	0
Application 13	120	0	1	100
Application 14	120	0	1	100
Application 15	120	50	0	0
Online Form Creation Software				
Application 16	120	0	1	100
Application 17	120	0	1	100
Application 18	120	80	0	0
Application 19	120	0	1	100
Application 20	120	0	1	100

Table 4.20 Reliability Test Evaluation (cont'd)

Application Name	Testing Time, t (s)	Failure Rate (λ)	Score	Reliability Score (%)
Company Web applications				
Application 21	120	0	1	100
Application 22	120	120	0	0
Application 23	120	130	0	0
Application 24	120	0	1	100
Application 25	120	0	1	100
Document Creation Software				
Application 26	120	0	1	100
Application 27	120	0	1	100
Application 28	120	0	1	100

Table 4.20 shows that under the Educational web application category, Applications 1, 4, and 5 performed well in the reliability test with no failure rate and had a Reliability score of 100% while Applications 2 and 3 failed the test with failure rates of 120 and 130 respectively. This shows that Applications 1, 4, and 5 can perform well within a specified time frame without encountering errors. The average Reliability score was 60%.

All the applications under the Video editing category had a Reliability score of 100% with no failure rate when tested for 120 seconds. The average Reliability score was 100%.

Under the e-commerce category, Applications 11, 13, and 14 had 100% as the Reliability score with no failure rate while Applications 12 and 15 failed the test with 70 and 50 as the failure rates, respectively. The average Reliability score was 60%.

Web applications under the online form creation software also performed well in the reliability test. Applications 16, 17, 19, and 20 had 100% as the Reliability score with no failure rate. Application 18 failed the test with 80 as the failure rate. Although applications under the category are seen to be reliable, they had an average Reliability score of 80%.

Under the company web application category, three (3) applications are seen to have performed well in the test while two (2) are seen to have performed poorly. Applications

21, 24, and 25 had 100% as the Reliability score with no failure rate while Applications 22 and 23 had 120 and 130 as the failure rates, respectively. The average Reliability score was also 60%.

Lastly, applications under the Document Creation Software category had 100% as the Reliability score and had no failure rate. The average Reliability score was 100% and the applications can be said to have performed well within a specified time frame without encountering errors.

The web application categories with the highest average Reliability score were the Video and Photo applications and Document Creation Software with the score of 100% while the ones with the lowest average scores were the Educational, E-commerce, and Company web applications with the score of 60%.

4.4.4 Software Testability

The web applications were evaluated for testability and the result is shown in Table 4.21. The python script conducted the test by simulating 500 concurrent users on each of the web applications. The value for the constraints used was from the software throughput values performed in the Efficiency test.

Table 4.21 Web Applications and Testability Test Rank

Application Name	Constraints (x)	Testing Time, t (sec)	Failure (λ)	ε_r (%)	Success Score	Success Rate (%)
Educational Web Applications						
Application 1	236.25	120	0	1	1	100
Application 2	178.56	120	0	1	1	100
Application 3	206.61	120	0	1	1	100
Application 4	165.59	120	20	1	0	0
Application 5	131.58	120	80	1	0	0
Video editing Web applications						
Application 6	136.37	120	0	1	1	100
Application 7	172.61	120	0	1	1	100
Application 8	156.72	120	0	1	1	100
Application 9	154.46	120	0	1	1	100
Application 10	137.11	120	0	1	1	100
E-commerce Software						
Application 11	224.17	120	0	1	1	100
Application 12	158.08	120	70	1	0	0
Application 13	177.642	120	0	1	1	100
Application 14	187.52	120	0	1	1	100
Application 15	161.67	120	50	1	0	0

Table 4.21 Web Applications and Testability Test Rank (cont'd)

Application Name	Constraints (x)	Testing Time, t (sec)	Failure (λ)	ε_r (%)	Success Score	Success Rate (%)
Online Form Creation Software						
Application 16	177.35	120	0	1	1	100
Application 17	162.93	120	0	1	1	100
Application 18	148.79	120	80	1	0	0
Application 19	140.95	120	0	1	1	100
Application 20	153.77	120	0	1	1	100
Company Web applications						
Application 21	176.08	120	0	1	1	100
Application 22	149.63	120	120	1	0	0
Application 23	149.38	120	130	1	0	0
Application 24	188.05	120	0	1	1	100
Application 25	147.18	120	0	1	1	100
Document Creation Software						
Application 26	138.15	120	0	1	1	100
Application 27	146.49	120	0	1	1	100
Application 28	95.65	120	0	1	1	100

On conducting a Testability test for the Educational web applications, it was noted that Applications 4 and 5 failed the test with 20 and 80 as the failure rate, respectively, while Applications 1, 2, and 3 passed the test with no failure rate when tested for 120 seconds. The average Testability score was seen to be 60%.

The Video editing software performed well in the Testability test with a success rate of 100% with no failure. The average Testability score was 100% and the correctness of the software can be said to have been verified.

The e-commerce software also had an average Testability score of 60% with three (3) Applications passing the test and the remaining two (2) failing the test. Applications 11, 13, and 14 had Testability scores of 100% with no failure rate while Applications 12 and 15 failed the test with 70 and 50 failure rates, respectively.

The Online form creation software had an average score of 80% with four (4) Applications passing the test and one (1) failing the test. Applications 16, 17, 19, and 20 had 100% as the Testability score while Application 18 had a failure rate of 80, hence, failed the test.

The Company web application category had an average Testability score of 60%. Applications 21, 24, and 25 had 100% as the Testability score with no encountered failure while Applications 22 and 23 had 0% as the Testability score with failure rates of 120 and 130, respectively.

Finally, the Document Creation Software performed well in the test with an average Testability score of 100% with no encountered failure rate. The correctness of the software can be said to have been verified with no encountered error.

In conclusion, the web application categories with the highest average Testability score were the Video and Photo applications and Document Creation Software with the score of 100% while the ones with the lowest average scores were the Educational, E-commerce, and Company web applications with the score of 60%.

4.4.5 Software Usability Test

The usability test adopted in this study was assessing the external usability factors. This was selected based on evaluating user experience on the web applications. To evaluate the Usability of the applications, a questionnaire was administered to individuals in both Nigeria and Ghana via their email. To get the highest the response rate, the people were guaranteed that their responses and identities would be treated with the highest confidentiality since the research was being conducted for academic purposes. Therefore, 100 correspondents were identified and issued the survey questionnaire and the rate of response rate was 65%. To evaluate the consistency of the survey scores, reliability of the score was calculated using Cronbach's Alpha model in IBM SPSS Statistics 26.0 software.

A summary showing the number of users who were administered the survey is shown in Table 4.22 while Table 4.23 shows the reliability statistics of the scores from the survey.

Table 4.22 Case Processing Summary for Survey

Survey Processing Survey		Number	%
Cases	Valid	64	98.5
	Excluded[a]	1	1.5
	Total	65	100.0

Excluded[a] is the listwise deletion based on all variables in the procedure.

Table 4.22 shows the entire respondents and the percentage score. The Cronbach's Alpha model gave a listwise deletion value is 1, indicating that, there was one redundant value. This was excluded from the model during the evaluation process. The percentage of valid values that were used for the evaluation was 98.5% while the excluded values were 1.5%.

The scores from the survey were evaluated to assess the reliability of the scores from the questionnaire using equation (4.6).

$$\alpha = N\bar{c}/(\bar{v} + (N - 1) * \bar{c}) \qquad (4.6)$$

where, α = Cronbach's Alpha;

N = Number of objects being considered;

$\bar{c}$ = Covariance between the considered objects; and

$\bar{v}$ = average variance.

Survey scores for applications that had Cronbach Alpha values less than 0.80 were said to be unreliable and sent for re-evaluation while values greater than 0.80 were accepted and tested for usability.

Table 4.23 Reliability Statistics of Survey Score

Application Name	**Cronbach's Alpha**
Educational Web Applications	
Application 1	0.964
Application 2	0.890
Application 3	0.902
Application 4	0.974
Application 5	0.864
Video editing Web applications	
Application 6	0.819
Application 7	0.829
Application 8	0.956
Application 9	0.921
Application 10	0.851
E-commerce Software	
Application 11	0.983
Application 12	0.852
Application 13	0.820
Application 14	0.910
Application 15	0.905
Online Form Creation Software	
Application 16	0.927
Application 17	0.935
Application 18	0.951
Application 19	0.924
Application 20	0.929

Table 4.23 Reliability Statistics of Survey Score

Application Name	**Cronbach's Alpha**
Company Web applications	
Application 21	0.915
Application 22	0.821
Application 23	0.847
Application 24	0.913
Application 25	0.879
Document Creation Software	
Application 26	0.957
Application 27	0.929
Application 28	0.954
Number of Items = 28	

Table 4.23 shows that from the survey for Educational web applications, Application 1 had Cronbach Alpha's value of 0.964, Application 2 had a Cronbach alpha value of 0.890, Application 3 had 0.902, Application 4 had 0.974 and Application 5 had 0.864. The reliability survey of the applications is seen to be within the Reliable and Very Reliable categories.

With the Video editing software, Application 6 had 0.819 as the Cronbach Alpha value, Application 7 had 0.829, Application 8 had 0.956, Application 9 had 0.921 while Application 10 also had 0.851 as Cronbach Alpha value. The values are also seen to be within the Reliable and Very Reliable categories.

The survey for E-commerce software recorded a Cronbach Alpha value of 0.983 for Application 11, 0.852 for Application 12, 0.820 for Application 13, 0.910 for Application 14, and 0.905 for Application 15. These values are also seen to be within the Reliable and Very Reliable categories.

For the Online form creation software, Application 16 had 0.927 as the Cronbach Alpha value, Application 17 had 0.935, Application 18 had 0.951, Application 19 had 0.924 while Application 20 also had 0.929. The values are all seen to be Very Reliable.

The Cronbach Alpha values for the Company web applications were seen to be within the Reliable and Very Reliable categories. Application 21 had Cronbach Alpha value of 0.915, Application 22 had 0.821, Application 23 had 0.847, Application 24 had 0.913 and Application 25 had 0.879.

Finally, the Cronbach Alpha values of the Document Creation Software were seen to be Very Reliable. Application 26 had the value of 0.957, Application 27 had 0.929 while Application 28 had 0.954.

Usability assessment was carried out on each of the web applications by using the scores obtained from the questionnaire. The total score was multiplied with the provided scale in order to get the total score.

Usability Evaluation of *Application 1*: The score of Application 1 was calculated by evaluating the survey statistics to get the total score as shown in Table 4.24.

Table 4.24 Survey Statistics for Application 1

Question	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	15	34	9	2	5
Easy Information search	20	28	11	0	6
Clearer Information organisation	17	30	11	1	6
Pleasant Interface	14	28	11	3	9
Usefulness of presented Images	15	34	9	1	6
Right Presentation of content	14	28	14	0	9
Appropriate Size of web controls	15	24	18	2	6
Less Load time	18	22	10	7	8
Meet Expected Functions	14	21	22	2	6
Overall Satisfaction	14	27	14	2	8
Sum	156	276	129	20	69
Total Score	156×5 = 780	276×4 = 1104	129×3 = 387	20×2 = 40	69×1 = 69
Sum of Total Score = 2380					

Table 4.24 shows that 156 scores was attained by Strongly Agree, Agree had 276 scores, Undecided had 129 scores, Disagree had 20 scores while Strongly Disagree had 95 scores.

Usability = Total Score/Maximum Score × 100

Maximum Score = Number of Respondents × Number of Survey × 5

= 65 × 10 × 5

= 3250

Usability = 2380/3250 × 100

=73.2

The percentage value of Application 1 lies in the *Good* class of the SUS score since it had a usa. This shows that there was an easy usage of the web application.

Usability Evaluation for *Application 2:* The score of Application 2 was calculated by evaluating the survey statistics to get the total score as shown in Table 4.25.

Table 4.25 Survey Statistics for Application 2

Question	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	22	30	11	2	0
Easy Information search	17	30	11	5	2
Clearer Information organisation	16	33	9	7	0
Pleasant Interface	15	21	18	6	5
Usefulness of presented Images	17	32	10	5	1
Right Presentation of content	17	34	11	2	1
Appropriate Size of web controls	16	26	18	4	0
Less Load time	15	24	17	7	2
Meet Expected Functions	12	21	19	8	5
Overall Satisfaction	14	20	14	6	1
Sum	161	271	138	52	17
Total Score	161×5 = 805	271×4 = 1084	138×3 = 414	52×2 = 104	17×1 = 17
Sum of Total Score = 2424					

Results from Table 4.25 indicate that, Strongly Agree had 161of the score, Agree had 271 of the score, Undecided had 138 of the score, while Disagree and Strongly Disagree had 52 and 17 respectively.

Usability = Total Score/Maximum Score × 100

Maximum Score = Number of Respondents × Number of Surveys × 5

= 65 × 10 × 5

=3250

Usability= 2424/3250 × 100

= 74.6

The percentage value of Application 2 based on the survey scores was 74.6%. This figure lies in the *Good* category of the SUS scale. It can be concluded that the web application can be easily navigated and learned by users.

Usability Evaluation for Application 3: The score of Application 3 was calculated by evaluating the survey statistics to get the total score as shown in Table 4.26.

Table 4.26 Survey Statistics for Application 3

Question	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	25	26	5	0	9
Easy Information search	24	26	6	1	8
Clearer Information organisation	19	32	6	0	8
Pleasant Interface	18	29	9	1	8
Usefulness of presented Images	17	26	10	2	10
Right Presentation of content	18	30	9	0	8
Appropriate Size of web controls	16	25	13	2	9
Less Load time	13	22	12	8	10
Meet Expected Functions	14	20	21	1	9
Overall Satisfaction	20	28	6	2	9
Sum	184	264	97	17	88
Total Score	184×5 = 920	264×4 = 1056	97×3 = 291	17×2 = 34	88×1 = 88
Sum of Total Score = 2389					

Results from the Table 4.26 show that, Strongly Agree had the value of 184 out of the overall score, Agree had the value of 264 out of the overall score, Undecided had the value of 97 out of the overall score while Disagree and Strongly Disagree had the values of 17 and 88 out of the overall score.

Usability = 2389/3250 × 100

=73.5

The percentage value of Application 3 was 73.5%. The results show that the application is in the *Good* category of the SUS scale.

Usability Evaluation for Application 4: Usability assessment of Application 4 was performed and the result is shown in Table 4.27.

Table 4.27 Survey Statistics for Application 4

Question	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	29	25	1	0	10
Easy Information search	30	23	3	0	9
Clearer Information organisation	29	22	4	0	10
Pleasant Interface	26	23	6	1	9
Usefulness of presented Images	22	27	5	2	9
Right Presentation of content	21	24	9	0	11
Appropriate Size of web controls	22	25	9	0	9
Less Load time	24	22	5	4	10
Meet Expected Functions	23	18	15	0	9
Overall Satisfaction	30	23	2	1	9
Sum	256	232	59	8	95
Total Score	256×5 = 1280	232×4 = 928	59×3 = 177	8×2 = 16	95×1 = 95
Sum of Total Score = 2496					

Based on Table 4.27, it can be said that the scores for Strongly Agree are 256, scores for Agree are 232, scores for Undecided are 59, scores for Disagree are 8, and scores for Strongly Disagree are 95.

Usability=2496/3250 × 100

=76.8

The results indicate that the percentage value of Application 4 is 76.8%. It can be concluded based on the analysis that the application was pleasant and met the user's satisfaction.

Usability Evaluation for Application 5: Usability assessment of Application 5 was performed and the result is shown in Table 4.28.

Table 4.28 Survey Statistics for Application 5

Question	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	23	29	11	2	0
Easy Information search	17	29	12	5	2
Clearer Information organisation	15	33	9	8	0
Pleasant Interface	14	22	19	5	5
Usefulness of presented Images	16	32	10	6	1
Right Presentation of content	17	34	11	2	1
Appropriate Size of web controls	15	26	18	6	0
Less Load time	14	24	17	8	2
Meet Expected Functions	12	22	22	8	1
Overall Satisfaction	14	29	15	6	1
Sum	157	280	144	56	13
Total Score	157×5 = 785	280×4 = 1120	144×3 = 432	56×2 = 112	13×1 = 13
Sum of Total Score = 2462					

Table 4.28 shows that, Strongly Agree had 157, Agree had 280, Undecided, Disagree and Strongly Disagree had 144, 56 and 13 respectively.

Usability=2462/3250 × 100

=75.75

The percentage value of Application 5 according on the survey score was 75.75%. This figure, which falls in the *Good* category of the SUS score, shows that the web application is learnable.

Usability Evaluation for *Application 6:* Usability assessment of Application 6 was performed and the result is shown in Table 4.29.

Table 4.29 Survey Statistics for Application 6

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	25	12	15	1	12
Easy Information search	29	15	14	5	2
Clearer Information organisation	27	22	6	10	0
Pleasant Interface	29	5	21	4	6
Usefulness of presented Images	21	18	17	7	2
Right Presentation of content	32	18	15	0	0
Appropriate Size of web controls	15	25	19	5	1
Less Load time	14	18	17	16	0
Meet Expected Functions	12	21	18	5	9
Overall Satisfaction	27	25	7	3	3
Sum	231	179	149	56	35
Total Score	231×5 = 1155	179×4 = 716	149×3 = 447	56×2 = 112	35×1 = 35
Sum of Total Score = 2465					

Table 4.29 displays the scores for Strongly Agree, Agree, Undecided, Disagree, and Strongly Disagree.

Usability=2465/3250 × 100

=75.84

The percentage value of Application 6 according to the survey was 75.84%. This value lies in the *Good* category of the SUS score. This shows that the website can be easily accessed by users.

Usability Evaluation for *Application 7:* Usability assessment of Application 7 was performed and the result is shown in Table 4.30.

Table 4.30 Survey Statistics for Application 7

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	20	15	12	13	5
Easy Information search	12	7	19	15	12
Clearer Information organisation	24	19	14	6	2
Pleasant Interface	30	12	19	0	4
Usefulness of presented Images	25	10	15	10	5
Right Presentation of content	27	12	14	12	0
Appropriate Size of web controls	15	13	17	9	11
Less Load time	18	16	15	0	16
Meet Expected Functions	12	25	16	6	6
Overall Satisfaction	23	22	17	0	3
Sum	206	151	158	71	64
Total Score	206×5 = 1030	151×4 = 604	158×3 = 474	71×2 = 142	64×1 = 64
Sum of Total Score = 2314					

According to Table 4.30, Strongly Agree had the score of 206, Agree had the score of 151, Undecided had the score of 158, Disagree had the score of 71, and Strongly Disagree had the score of 64.

Usability=2314/3250 × 100

=71.20

The percentage value of Application 7 is 71.20%. This falls in the *Good* category of the SUS score.

Usability Evaluation for *Application 8:* Usability assessment of Application 8 was performed and the result is shown in Table 4.31.

Table 4.31 Survey Statistics for Application 8

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	25	14	12	14	0
Easy Information search	16	9	15	13	12
Clearer Information organisation	29	17	13	5	1
Pleasant Interface	26	15	18	2	4
Usefulness of presented Images	21	14	17	8	5
Right Presentation of content	23	18	13	9	2
Appropriate Size of web controls	32	16	17	0	0
Less Load time	29	16	16	0	4
Meet Expected Functions	25	22	12	0	6
Overall Satisfaction	20	22	17	3	3
Sum	246	163	150	54	37
Total Score	246×5 = 1230	163×4 = 652	150×3 = 450	54×2 = 108	37×1 = 37
Sum of Total Score = 2477					

Table 4.31 shows that Strongly Agree had the score of 246, Agree had the score of 163, Undecided had the score of 150, Disagree had the score of 54, and Strongly Disagree had the score of 37.

Usability=2477/3250 × 100

=76.21

The percentage value of Application 8 was evaluated as 76.21%. This value is in the *Good* category of the SUS scale.

Usability Evaluation for *Application 9:* Usability assessment of Application 9 was performed and the result is shown in Table 4.32.

Table 4.32 Survey Statistics for Application 9

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	29	16	15	5	0
Easy Information search	25	17	13	5	5
Clearer Information organisation	29	14	12	9	1
Pleasant Interface	28	12	14	9	2
Usefulness of presented Images	23	18	19	5	0
Right Presentation of content	12	29	16	8	0
Appropriate Size of web controls	19	19	16	6	5
Less Load time	35	19	2	5	4
Meet Expected Functions	28	12	16	0	9
Overall Satisfaction	22	20	3	17	3
Sum	250	176	126	69	29
Total Score	250×5 = 1250	176×4 = 704	126×3 = 378	69×2 = 138	29×1 = 29
Sum of Total Score = 2499					

According to Table 4.32, it can be said that the scores for Strongly Agree are 250, scores for Agree are 176, scores for Undecided are 126, scores for Disagree are 69, and scores for Strongly Disagree are 29.

Usability=2499/3250 × 100

=76.89

The percentage value of Application 9 was evaluated as 76.89%. This value shows that the website can be easily learned and accessed by users.

Usability Evaluation for *Application 10:* Usability assessment of Application 10 was performed and the result is shown in Table 4.33.

Table 4.33 Survey Statistics for Application 10

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	24	19	14	7	1
Easy Information search	29	15	12	8	1
Clearer Information organisation	26	19	10	9	1
Pleasant Interface	26	12	16	9	2
Usefulness of presented Images	29	16	17	3	0
Right Presentation of content	14	25	19	2	5
Appropriate Size of web controls	26	12	19	3	5
Less Load time	29	21	6	9	0
Meet Expected Functions	28	12	14	4	7
Overall Satisfaction	26	18	3	12	6
Sum	257	169	130	66	28
Total Score	257×5 = 1285	169×4 = 676	130×3 = 390	66×2 = 132	28×1 = 28
Sum of Total Score = 2511					

Table 4.33 shows that the value for Strongly Agree was 257, value for Agree was 169, value for Undecided was 130, value for Disagree was 66, and value for Strongly Disagree was 28.

Usability=2511/3250 × 100

=77.26

The percentage value of Application10 was 77.26%. This falls in the *Good* category of the SUS score. The attained result shows that the application could be easily navigated and had a clearer organisation of information.

Usability Evaluation for *Application 11:* Usability assessment of Application 11 was performed and the result is shown in Table 4.34.

Table 4.34 Survey Statistics for Application 11

Question	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	20	18	9	15	3
Easy Information search	15	24	12	5	9
Clearer Information organisation	15	23	9	6	12
Pleasant Interface	18	27	10	3	7
Usefulness of presented Images	16	25	10	6	8
Right Presentation of content	13	30	15	7	0
Appropriate Size of web controls	15	15	18	12	5
Less Load time	14	22	15	5	9
Meet Expected Functions	19	18	15	0	13
Overall Satisfaction	17	19	15	6	8
Sum	162	221	128	65	74
Total Score	162×5 = 810	221×4 = 884	128×3 = 384	65×2 = 130	74×1 = 74
Sum of Total Score = 2282					

According to Table 4.34, Strongly Agree had 162, Agree had 221, Undecided had 128, Disagree had 65, and Strongly Disagree had 74.

Usability=2282/3250 × 100

=70.21

The percentage value of Application 11 was evaluated as 70.21%. This figure lies in the *Good* category of the SUS score.

Usability Evaluation for Application 12: Usability assessment of Application 12 was performed and the result is shown in Table 4.35.

Table 4.35 Survey Statistics for Application 12

Question	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	18	20	0	15	12
Easy Information search	10	24	16	8	7
Clearer Information organisation	12	21	12	8	12
Pleasant Interface	18	27	10	3	7
Usefulness of presented Images	13	22	8	10	12
Right Presentation of content	19	27	12	0	7
Appropriate Size of web controls	12	14	0	19	20
Less Load time	17	21	12	7	8
Meet Expected Functions	15	20	12	0	18
Overall Satisfaction	18	14	12	8	13
Sum	152	210	94	78	116
Total Score	152×5 = 760	210×4 = 840	94×3 = 282	78×2 = 156	116×1 = 116
Sum of Total Score = 2154					

Table 4.35 shows that the value for Strongly Agree is 152, value for Agree is 210, value for Undecided is 94, value for Disagree is 78, and value for Strongly Disagree is 116.

Usability=2154/3250 × 100

= 66.27

The percentage value of Application 12 was further evaluated as 66.27%. This value lies in the *Poor* category of the SUS score. It can be concluded that Application was difficult to navigate due to a number of non-working functions.

Usability Evaluation for *Application 13:* The Usability evaluation for Application 13 was done and results were shown in Table 4.36.

Table 4.36 Survey Statistics for Application 13

Question	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	13	14	0	18	20
Easy Information search	15	27	18	0	5
Clearer Information organisation	22	25	0	5	13
Pleasant Interface	12	10	24	19	0
Usefulness of presented Images	13	22	8	10	12
Right Presentation of content	28	15	1	20	1
Appropriate Size of web controls	8	25	5	12	15
Less Load time	19	15	12	12	7
Meet Expected Functions	13	18	12	6	16
Overall Satisfaction	12	16	13	9	15
Sum	155	187	93	111	114
Total Score	155×5 = 775	187×4 = 748	93×3 = 279	111×2 = 222	114×1 = 114
Sum of Total Score = 2128					

Table 4.36 shows the scores attained by Strongly Agree, Agree, Undecided, Disagree and Strongly Disagree. Strongly Agree had 155 out of the survey, Agree had 187 out of the survey, Undecided had 93, Disagree Strongly had 111while Disagree had 114.

Usability=2128/3250 × 100

= 65.47

The percentage value of Application 13 according to the values from the evaluation was 65.47%. This falls in the *Poor* category of the SUS score.

Usability Evaluation for *Application 14:* Usability evaluation for Application 14 was done and results were shown in Table 4.37.

Table 4.37 Survey Statistics for Application 14

Question	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	19	16	11	18	1
Easy Information search	25	22	16	2	0
Clearer Information organisation	27	21	10	5	2
Pleasant Interface	19	24	14	7	1
Usefulness of presented Images	23	22	8	10	2
Right Presentation of content	28	18	9	8	2
Appropriate Size of web controls	18	15	9	18	5
Less Load time	18	12	18	10	7
Meet Expected Functions	15	19	17	6	8
Overall Satisfaction	28	19	0	3	15
Sum	220	188	112	87	43
Total Score	220×5 = 1100	188×4 = 752	112×3 = 336	87×2 = 174	43×1 = 43
Sum of Total Score = 2405					

According to Table 4.37, Strongly Agree had 220 out of the scores, Agree had 188 out of the scores, Undecided had 112 out of the scores, Disagree had 87 out of the scores, and Strongly Disagree had 43 out of the scores.

Usability=2405/3250 × 100

=74.0

The percentage value of Application 14 was evaluated as 74.0%. This falls in the *Good* category of the SUS score. This shows that the website had a lesser load time.

Usability Evaluation for *Application 15:* Usability assessment for Application 15 was done and results were shown in Table 4.38.

Table 4.38 Survey Statistics for Application 15

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	18	15	12	0	20
Easy Information search	24	21	20	0	0
Clearer Information organisation	29	23	5	5	3
Pleasant Interface	25	21	12	6	1
Usefulness of presented Images	29	16	10	8	2
Right Presentation of content	28	15	8	10	4
Appropriate Size of web controls	28	20	11	3	3
Less Load time	19	16	14	11	5
Meet Expected Functions	12	25	15	6	7
Overall Satisfaction	28	25	7	6	2
Sum	240	197	114	55	47
Total Score	240×5 = 1200	197×4 = 788	114×3 = 342	55×2 = 110	47×1 = 47
Sum of Total Score = 2487					

According to Table 4.38, Strongly Agree had 240, Agree had 197, Undecided had 114, Disagree had 55, while Strongly Disagree had 47.

Usability=2487/3250 × 100

=76.52

The percentage value of Application 15 was evaluated as 76.52%. This falls in the *Good* category of the SUS score. This shows that the website can be easily used since it had a pleasant interface with useful images.

Usability Evaluation for *Application 16:* Usability assessment for Application 16 was done and results were shown in Table 4.39.

Table 4.39 Survey Statistics for Application 16

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	30	17	12	5	1
Easy Information search	29	16	12	7	1
Clearer Information organisation	28	18	13	5	1
Pleasant Interface	22	15	16	10	2
Usefulness of presented Images	28	16	19	2	0
Right Presentation of content	25	14	2	19	5
Appropriate Size of web controls	28	14	15	5	3
Less Load time	27	23	9	6	0
Meet Expected Functions	27	15	12	5	6
Overall Satisfaction	26	18	12	6	3
Sum	270	166	122	70	22
Total Score	270×5 = 1350	166×4 = 664	122×3 = 366	70×2 = 140	22×1 = 22
Sum of Total Score = 2542					

According to Table 4.39, Strongly Agree is seen to have scores of 270, Agree is seen to have scores of 166, Undecided is seen to have scores of 122, Disagree is seen to have scores of 70, while Strongly Disagree is also seen to have scores of 22.

Usability=2542/3250 × 100

=78.21

The percentage value of Application 16 was calculated as 78.21%. This falls in the *Good* category of the SUS score. This shows that the website the users expected functions.

Usability Evaluation for *Application 17:* Usability assessment for Application 17 was done and results were shown in Table 4.40.

Table 4.40 Survey Statistics for Application 17

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	25	15	12	1	12
Easy Information search	31	19	12	3	0
Clearer Information organisation	26	22	7	10	0
Pleasant Interface	25	15	15	4	6
Usefulness of presented Images	21	17	18	7	2
Right Presentation of content	32	15	15	3	0
Appropriate Size of web controls	30	15	14	5	1
Less Load time	20	10	18	17	0
Meet Expected Functions	16	15	18	8	8
Overall Satisfaction	28	22	3	7	5
Sum	254	165	132	65	34
Total Score	254×5 = 1270	165×4 = 660	132×3 = 396	65×2 = 130	34×1 = 34
Sum of Total Score = 2490					

According to Table 4.40, it is seen that Strongly Agree had 254, Agree had 165, Undecided had 132, Disagree had 65, and Strongly Disagree had 34.

Usability=2490/3250 × 100

=76.61

The percentage value of Application 17 according to survey score was 76.61%. This falls in the *Good* category of the SUS score. This shows that the website had a pleasant user interface.

Usability Evaluation for *Application 18:* Usability assessment for Application 18 was done and results were shown in Table 4.41.

Table 4.41 Survey Statistics for Application 18

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	29	10	12	5	9
Easy Information search	36	15	10	3	1
Clearer Information organisation	29	17	13	6	0
Pleasant Interface	24	16	15	4	6
Usefulness of presented Images	24	19	13	7	2
Right Presentation of content	29	18	17	1	0
Appropriate Size of web controls	26	16	13	9	1
Less Load time	23	8	17	16	1
Meet Expected Functions	21	12	18	14	0
Overall Satisfaction	25	23	8	5	4
Sum	266	154	136	70	24
Total Score	266×5 = 1330	154×4 = 616	136×3 = 408	70×2 = 140	24×1 = 24
Sum of Total Score = 2518					

Table 4.41 shows that, Strongly Agree had 266, Agree had 154, Undecided had 136, Disagree had 70, and Strongly Disagree also had 24.

Usability=2518/3250 × 100

=77.47

The percentage value of Application 18 according to the survey scores was 77.47%. This falls in the *Good* category of the SUS score. This shows that the application met the expected functions of users.

Usability Evaluation for *Application 19:* Usability assessment for Application 19 was done and results were shown in Table 4.42.

Table 4.42 Survey Statistics for Application 19

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	23	13	15	9	5
Easy Information search	26	19	16	3	1
Clearer Information organisation	27	16	19	1	2
Pleasant Interface	27	19	17	0	2
Usefulness of presented Images	22	16	19	2	6
Right Presentation of content	24	19	13	8	1
Appropriate Size of web controls	26	20	12	0	7
Less Load time	29	7	14	14	1
Meet Expected Functions	18	6	16	19	6
Overall Satisfaction	27	23	12	1	2
Sum	249	158	153	57	33
Total Score	249×5 = 1245	158×4 = 632	153×3 = 459	57×2 = 114	33×1 = 33
Sum of Total Score = 2483					

Table 4.42 shows that, Strongly Agree had 249 out of the total scores, Agree had 158 out of the total scores, Undecided had 153 out of the total scores, Disagree had 57 out of the total scores, and Strongly Disagree had 33 out of the total scores.

Usability=2483/3250 × 100

=76.40

The percentage value of Application 19 was evaluated as 76.40%. This falls in the *Good* category of the SUS score. This shows that content on the website were rightly presented and in an orderly manner.

Usability Evaluation for *Application 20:* Usability assessment for Application 20 was done and results were shown in Table 4.43.

Table 4.43 Survey Statistics for Application 20

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	29	12	13	7	4
Easy Information search	23	15	19	3	5
Clearer Information organisation	27	16	19	1	2
Pleasant Interface	25	12	19	6	3
Usefulness of presented Images	22	19	16	6	2
Right Presentation of content	24	19	8	13	1
Appropriate Size of web controls	18	3	20	15	9
Less Load time	21	7	26	1	10
Meet Expected Functions	28	2	18	15	2
Overall Satisfaction	23	29	12	1	0
Sum	240	134	170	68	38
Total Score	240×5 = 1200	134×4 = 536	170×3 = 510	68×2 = 136	38×1 = 38
Sum of Total Score = 2420					

According to Table 4.43, Strongly Agree had 240, Agree had 134, Undecided had 170, Disagree had 68, and Strongly Disagree had 38.

Usability=2420/3250 × 100

=74.46

The percentage value of Application 20 according to the scores survey was evaluated as 74.46%. This falls in the *Good* category of the SUS score. This shows that there was an overall satisfaction of the website usage by the users.

Usability Evaluation for *Application 21:* Usability assessment for Application 21 was done and results were shown in Table 4.44.

Table 4.44 Survey Statistics for Application 21

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	29	15	18	1	2
Easy Information search	15	13	25	5	7
Clearer Information organisation	25	16	14	1	9
Pleasant Interface	18	19	17	6	5
Usefulness of presented Images	26	12	19	2	6
Right Presentation of content	27	16	13	8	1
Appropriate Size of web controls	22	12	13	10	8
Less Load time	15	19	13	16	2
Meet Expected Functions	27	5	16	16	1
Overall Satisfaction	25	26	12	0	2
Sum	229	153	160	65	43
Total Score	229×5 = 1145	153×4 = 612	160×3 = 480	65×2 = 130	43×1 = 43
Sum of Total Score = 2410					

Table 4.44 shows that, Strongly Agree had 229 out of the survey scores, Agree had 153 out of the survey scores, Undecided had 160 out of the survey scores, Disagree had 65 out of the survey scores, and Strongly Disagree had 43 out of the survey scores.

Usability=2410/3250 × 100

=74.15

The percentage value of Application 21 was evaluated as 74.15%. This falls in the *Good* category of the SUS score.

Usability Evaluation for *Application 22:* Usability assessment for Application 22 was done and results were shown in Table 4.45.

Table 4.45 Survey Statistics for Application 22

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	31	12	15	6	1
Easy Information search	17	15	22	8	3
Clearer Information organisation	23	18	12	6	6
Pleasant Interface	25	13	16	8	3
Usefulness of presented Images	26	12	19	2	6
Right Presentation of content	27	19	11	5	3
Appropriate Size of web controls	25	16	8	7	9
Less Load time	28	12	17	6	2
Meet Expected Functions	27	15	13	9	1
Overall Satisfaction	28	25	8	1	3
Sum	257	157	141	58	37
Total Score	257×5 = 1285	157×4 = 628	141×3 = 423	58×2 = 116	37×1 = 37
Sum of Total Score = 2489					

Table 4.45 indicates that, Strongly Agree had 257, Agree had 157, Undecided had 141, Disagree had 58, and scores for Strongly Disagree are 37.

Usability=2489/3250 × 100

=76.58

The percentage value of Application 22 according to the scores was seen to be 76.58%.

Usability Evaluation for *Application 23:* Usability assessment for Application 23 was done and results were shown in Table 4.46.

Table 4.46 Survey Statistics for Application 23

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	35	15	9	5	1
Easy Information search	20	15	20	7	3
Clearer Information organisation	25	16	6	12	6
Pleasant Interface	28	16	13	5	3
Usefulness of presented Images	26	19	12	6	2
Right Presentation of content	19	25	14	4	3
Appropriate Size of web controls	23	14	12	9	7
Less Load time	25	17	15	5	3
Meet Expected Functions	24	18	16	2	5
Overall Satisfaction	31	26	5	1	2
Sum	256	181	122	56	35
Total Score	256×5 = 1280	181×4 = 724	122×3 = 366	56×2 = 112	35×1 = 35
Sum of Total Score = 2517					

Table 4.46 displays that, Strongly Agree had 256, Agree had 181, Undecided had 122, Disagree had 56, and Strongly Disagree had 35.

Usability=2517/3250 × 100

=77.44

The percentage value of Application 23 according to the survey scores was evaluated as 77.44%. This falls in the *Good* category of the SUS score. This shows that there was an appropriate size of web controls on the web application.

Usability Evaluation for *Application 24:* Usability assessment for Application 24 was done and results were shown in Table 4.47.

Table 4.47 Survey Statistics for Application 24

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	32	18	5	8	2
Easy Information search	22	14	19	10	0
Clearer Information organisation	27	14	8	9	7
Pleasant Interface	30	12	8	10	5
Usefulness of presented Images	31	17	14	0	3
Right Presentation of content	21	19	16	9	0
Appropriate Size of web controls	29	16	16	0	4
Less Load time	16	25	16	3	5
Meet Expected Functions	23	20	15	4	3
Overall Satisfaction	29	14	5	15	2
Sum	260	169	122	68	31
Total Score	260×5 = 1300	169×4 = 676	122×3 = 366	68×2 = 136	31×1 = 31
Sum of Total Score = 2509					

Table 4.47 shows that, Strongly Agree had 260, Agree had 169, Undecided had 122, Disagree had 68, and Strongly Disagree also had 31.

Usability=2509/3250 × 100

=77.20

The percentage value of Application 24 according to the survey scores was evaluated as 77.20%. This lies in the *Good* category of the SUS scale.

Usability Evaluation for *Application 25:* Usability assessment for Application 25 was done and results were shown in Table 4.48.

Table 4.48 Survey Statistics for Application 25

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	23	30	2	6	4
Easy Information search	24	14	17	8	2
Clearer Information organisation	29	12	10	8	6
Pleasant Interface	32	11	12	7	3
Usefulness of presented Images	31	19	9	2	4
Right Presentation of content	24	15	9	16	1
Appropriate Size of web controls	31	12	15	4	3
Less Load time	22	16	19	5	3
Meet Expected Functions	25	18	12	7	3
Overall Satisfaction	26	15	9	13	2
Sum	267	162	114	76	31
Total Score	267×5 = 1335	162×4 = 648	114×3 = 342	76×2 = 152	31×1 = 31
Sum of Total Score = 2508					

According to Table 4.48, the value attained by Strongly Agree was 267, value attained by Agree was 162, value attained by Undecided was 114, value attained by Disagree was 76, and lastly, value attained by Strongly Disagree was 31.

Usability=2508/3250 × 100

=77.16

The percentage value of Application 25 was 77.16%. This falls in the *Good* category of the SUS score.

Usability Evaluation for *Application 26:* Usability assessment for Application 26 was done and results were shown in Table 4.49.

Table 4.49 Survey Statistics for Application 26

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	30	23	4	2	6
Easy Information search	24	17	14	7	3
Clearer Information organisation	26	15	10	10	4
Pleasant Interface	31	15	8	9	2
Usefulness of presented Images	31	16	12	6	0
Right Presentation of content	29	14	12	9	1
Appropriate Size of web controls	31	15	12	7	0
Less Load time	22	21	5	14	3
Meet Expected Functions	27	15	15	5	3
Overall Satisfaction	26	13	12	8	6
Sum	277	164	104	77	28
Total Score	277×5 = 1385	164×4 = 656	104×3 = 312	77×2 = 154	28×1 = 28
Sum of Total Score = 2535					

Table 4.49 presents the scores attained during the evaluation. Strongly Agree had 277, Agree had 164, Undecided had 104, Disagree had 77, and Strongly Disagree had 28.

Usability=2535/3250 × 100

=78.0

The percentage value of Application 26 according to the survey scores was 78.0%. This falls in the *Good* category of the SUS score.

Usability Evaluation for *Application 27:* Usability assessment for Application 27 was done and results were shown in Table 4.50.

Table 4.50 Survey Statistics for Application 27

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	32	20	6	2	5
Easy Information search	24	18	7	13	3
Clearer Information organisation	29	12	5	15	4
Pleasant Interface	36	15	3	6	5
Usefulness of presented Images	32	20	11	2	0
Right Presentation of content	14	29	9	12	1
Appropriate Size of web controls	23	15	13	9	5
Less Load time	21	22	8	11	3
Meet Expected Functions	26	16	12	9	2
Overall Satisfaction	25	14	18	3	5
Sum	262	181	92	82	33
Total Score	262×5 = 1310	181×4 = 724	92×3 = 276	82×2 = 164	33×1 = 33
Sum of Total Score = 2507					

According to Table 4.50, the value attained by Strongly Agree was 262, value attained by Agree was 181, value attained by Undecided was 92, value attained by Disagree was 82, and finally, value attained by Strongly Disagree was 33.

Usability=2507/3250 × 100

=77.13

The percentage value of Application 27 was evaluated as 77.13%.

Usability Evaluation for *Application 28:* Usability assessment for Application 28 was done and results were shown in Table 4.51.

Table 4.51 Survey Statistics for Application 28

Questions	**Strongly Agree**	**Agree**	**Undecided**	**Disagree**	**Strongly Disagree**
Easy Navigation	29	19	9	5	3
Easy Information search	26	17	8	11	3
Clearer Information organisation	28	12	8	13	4
Pleasant Interface	29	13	14	5	4
Usefulness of presented Images	27	19	15	2	2
Right Presentation of content	29	15	9	11	1
Appropriate Size of web controls	24	16	12	8	5
Less Load time	22	21	11	5	6
Meet Expected Functions	28	14	13	8	2
Overall Satisfaction	29	13	15	3	5
Sum	271	159	114	71	35
Total Score	271×5 = 1355	159×4 = 636	114×3 = 342	71×2 = 142	35×1 = 35
Sum of Total Score = 2510					

According to Table 4.51, the value attained by Strongly Agree was 271, value attained by Agree was 159, value attained by Undecided was 114, value attained by Disagree was 71, and value attained by Strongly Disagree was also 35.

Usability=2510/ 3250 x 100

=77.23

The percentage value of Application 28 based on the scores from the survey was calculated as 77.23%. This falls in the *Good* category of the SUS score. This shows that the website can be easily understood and used by users.

The overall usability scale for evaluating the web applications is shown in Table 4.52.

Table 4.52 Overall Usability Evaluation

Application Name	Total Score	Maximum Score	Usability Scale (%)
Educational Web Applications			
Application 1	2380	3250	73.20
Application 2	2424	3250	74.60
Application 3	2389	3250	73.50
Application 4	2496	3250	76.80
Application 5	2462	3250	75.75
Video editing Web applications			
Application 6	2465	3250	75.84
Application 7	2314	3250	71.20
Application 8	2477	3250	76.21
Application 9	2499	3250	76.89
Application 10	2511	3250	77.26
E-commerce Software			
Application 11	2282	3250	70.21
Application 12	2154	3250	66.27
Application 13	2128	3250	65.47
Application 14	2405	3250	74.00
Application 15	2487	3250	76.52
Online Form Creation Software			
Application 16	2542	3250	78.21
Application 17	2490	3250	76.61
Application 18	2518	3250	77.47
Application 19	2483	3250	76.40
Application 20	2420	3250	74.46

Table 4.52 Overall Usability Evaluation (cont'd)

Application Name	Total Score	Maximum Score	Usability Scale (%)
Company Web applications			
Application 21	2410	3250	74.15
Application 22	2489	3250	76.58
Application 23	2517	3250	77.44
Application 24	2509	3250	77.20
Application 25	2508	3250	77.16
Document Creation Software			
Application 26	2535	3250	78.00
Application 27	2507	3250	77.13
Application 28	2510	3250	77.23

According to Table 4.52, web applications under the Educational category had Usability scores ranging from 76.8% to 73.2%. Application 1 had the lowest score of 73.2%, followed by Application 3 with 73.5%, Application 2 with 74.6%, Application 5 with 75.75% while Application 4 had the highest usability score of 76.80%. The average Usability score was 74.77%.

Under the Video editing category, the Usability score ranged between 71.20% to 77.26%. Application 6 had 75.84%, Application 7 had 71.20%, Application 8 had 76.21%, Application 9 had 76.89% while Application 10 had a usability score of 77.26%. The average Usability score was 75.48%.

Under the e-commerce category, Application 11 had 70.21%, Application 12 had 66.27%, Application 13 had 65.47%, Application 14 had 74.0% while Application 15 had 76.52%. Application 13 had the lowest usability score while Application 15 had the highest score. The average Usability score was 70.494%.

Application 16 under the online form creation software category had the highest Usability score with 78.21% while Application 20 had the lowest score with 74.46%. Application 17 had 76.61%, Application 18 had 77.47% while Application 19 had 76.40%. The average Usability score was 76.63%.

The company web applications had Usability scores ranging from 77.44% to 74.15%. Application 23 had the highest Usability score of 77.44% followed by Application 20 with 77.20%. Application 25 was third with 77.16%. Application 22 had 76.58% while Application 21 recorded the least score with 74.15% in the category. The average Usability score was 76.506%.

Lastly, the Document Creation Software had a Usability score ranging from 78.00% to 77.13%. Application 26 had the highest score of 78.00% followed by Application 28 with 77.23% and lastly, Application 27 had 77.13%. The average Usability score was 77.453%.

4.4.6 Software Maintainability Test

Maintainability evaluation was done by sending HTTP get requests to the web applications. The python script was automated to send HTTP requests to the website being evaluated at every 10 seconds for a period of 3600 seconds. The time the web applications were unavailable for usage was recorded as the total downtime and the number of times the unavailability occurred was recorded as the number of failures the software encountered.

The maintainability rate for the web applications is shown in Table 4.53.

Table 4.53 Software Maintainability Evaluation

Application Name	Total Downtime	Number of Failures	Maintainability Score	Maintainability Rate (%)
Educational Web Applications				
Application 1	0.010% or 1 min 20 secs	8	0.022	99.978
Application 2	0.031% or 5 min 4 secs	27	0.150	99.850
Application 3	0.013% or 1 min 59 secs	10	0.098	99.957
Application 4	0.016% or 2 min 45 secs	13	0.073	99.927
Application 5	0.043% or 5 min 36 sec	32	0.328	99.672
Video editing Web applications				
Application 6	0.004% or 25 secs	2	0.005	99.996
Application 7	0% or 0 min 5 sec	1	0.000	100
Application 8	0% or 0 min 5 sec	1	0.000	100
Application 9	0.005% or 40 secs	3	0.007	99.993
Application 10	0% or 0 min 5 sec	1	0.000	100
E-commerce Software				
Application 11	0.019% or 2 min 15 secs	15	0.073	99.927
Application 12	0% or 0 min 5 sec	1	0.000	100
Application 13	0% or 0 min 5 sec	1	0.000	100
Application 14	0.014% or 2 min 5 secs	11	0.055	99.945
Application 15	0% or 0 min 5 sec	1	0.000	100
Online Form Creation Software				
Application 16	0% or 0 min 5 sec	1	0.000	100
Application 17	0% or 0 min 5 sec	1	0.000	100
Application 18	0% or 0 min 5 sec	1	0.000	100
Application 19	0% or 0 min 5 sec	1	0.000	100
Application 20	0% or 0 min 5 sec	1	0.000	100

Table 4.53 Software Maintainability Evaluation (cont'd)

Application Name	Total Downtime	Number of Failures	Maintainability Score	Maintainability Rate (%)
Company Web applications				
Application 21	0.006% or 1 min 5 secs	5	0.019	99.981
Application 22	0.021% or 2 min 48 secs	17	0.098	99.902
Application 23	0.013% or 1 min 59 secs	10	0.098	99.957
Application 24	0.010% or 1 min 20 secs	8	0.022	99.978
Application 25	0.029% or 5 min 2 secs	25	0.110	99.890
Document Creation Software				
Application 26	0% or 0 min 5 sec	1	0.000	100
Application 27	0% or 0 min 5 sec	1	0.000	100
Application 28	0% or 0 min 5 sec	1	0.000	100

All web applications according to Table 4.53, are seen to have performed well in the maintainability test. The web applications under the Educational web application category are seen to have Maintainability scores of 0.022, 0.150, 0.098, 0.073, and 0.008. Application 1 performed well with a total downtime of 1 min 20 secs with 8 number of failures and a score of 0.022 representing 99.978%. Application 2 had a downtime of 5 mins 4 secs with 27 failures; its Maintainability rate was 99.850%. Application 5 recorded the highest downtime of 5 mins 36 sec with 32 failures. Application 3 equally performed well with a downtime of 1 min 59 secs and 10 failures. Application 4 also performed fairly well with 2 min 45 secs as its downtime with 13 failures. Although the web applications passed the maintainability test, Application 5 and Application 2 recorded higher failure rates.

Three (3) applications under the Photo and Video editing category recorded a Maintainability rate of 100%. Applications 7, 8, and 10 had downtimes of 5 sec with 1 failure and Maintainability scores of 0.00 each. Also, Application 6 had a downtime of 25 sec with 2 failures and a Maintainability rate of 99.996%. Lastly, Application 9 recorded the lowest Maintainability rate under the category with 99.993% and 40 sec as the total downtime with 3 failures.

The e-commerce software recorded three (3) applications with 1 failure and a Maintainability rate of 100%. These applications were Applications 12, 13, and 15. Application 11 had a downtime of 2 min 15 secs with 15 failures and a Maintainability rate of 99.927% while Application 14 had a Maintainability rate of 99.945% with a total downtime of 2 min 5 sec and 11 failures.

The online form creation and Document Creation Software had Maintainability rates of 100% with 1 failure each and downtimes of 5 sec. These applications can be said to be easily modified to correct faults.

Web applications under the company web application category had downtimes ranging between 5 min 2 sec and 1 min 5 sec. Application 25 recorded the highest downtime of 5 mins 2 sec with 25 failures and 99.890% as Maintainability rate. Application 22 was next with 2 mins 48 sec as its downtime and 17 failures. Application 23 also had a downtime of 1 min 59 sec with 10 failures. Application 24 performed well with a downtime of 1 min 20 secs and 8 failures while Application 21 recorded the smallest downtime under the category with 1 min 5 sec and 5 failures.

The average Maintainability rates under the Educational, Video editing, E-commerce, Online form creation, Company, and Document Creation Software categories were 99.8768%, 99.9978%, 99.9744%, 100%, 99.9416%, and 100%, respectively.

4.4.7 Software Portability Test

Software portability test was carried out to assess the ease of porting the web applications from one web browser to the other as shown in Table 4.54. The web browsers used for the test were Google Chrome version 89.0, Mozilla Firefox version 87.0, Microsoft Edge version 89.0, and Safari version 5.1.

The python script that was written allows the web applications to open in the listed web browsers and records an average of the time it took to open in each web browser. It also recorded the average speed with which the website opened in web browsers.

Acceleration was calculated by dividing the average speed over the average time.

Finally, rate of transfer was calculated by dividing the average time over the total number of web browsers used for the evaluation.

Table 4.54 Software Portability Evaluation

Application Name	Acceleration, a (m/s^2)	Time, t (sec)	Speed, v (m/s)	Rate of Transfer, P_o	Portability Score (%)
Educational Web Applications					
Application 1	0.550	10.0	5.50	60	92.50
Application 2	1.000	8.5	8.50	51	95.625
Application 3	0.611	9.0	5.50	54	99.00
Application 4	0.906	8.0	7.25	48	95.00
Application 5	0.722	9.0	6.50	54	96.75
Video editing Web applications					
Application 6	0.554	11.0	6.00	66	99.00
Application 7	0.619	10.5	6.50	63	99.75
Application 8	1.017	8.5	8.65	51	99.66
Application 9	0.495	11.5	5.69	69	99.99
Application 10	2.623	6.0	15.74	36	99.72
E-commerce Software					
Application 11	1.570	6.5	10.25	39	99.94
Application 12	1.538	6.5	10.00	39	91.00
Application 13	1.033	7.5	7.75	45	98.45
Application 14	0.765	8.5	6.5	51	95.63
Application 15	0.778	9	7	54	99.00
Online Form Creation Software					
Application 16	1.493	7.5	11.20	45	99.94
Application 17	1.250	8.0	10.00	48	100.00
Application 18	0.936	9.0	8.42	54	99.99
Application 19	0.853	9.5	8.1	57	99.75
Application 20	0.471	12.0	5.65	72	99.90

Table 4.54 Software Portability Evaluation (cont'd)

Application Name	Acceleration, a (m/s^2)	Time, t (sec)	Speed, v (m/s)	Rate of Transfer, P_o	Portability Score (%)
Company Web applications					
Application 21	0.889	9.0	8.00	54	99.00
Application 22	0.824	8.5	7.00	51	93.50
Application 23	0.906	8.0	7.25	48	91.00
Application 24	1.692	6.5	11.00	39	97.50
Application 25	1.500	7.0	10.50	42	99.75
Document Creation Software					
Application 26	3.400	5.0	17.00	30	100.00
Application 27	2.167	6.0	13.00	36	99.00
Application 28	0.822	9.7	8.01	58	99.98

All web applications performed well in the test, according to Table 4.53, with a Portability score ranging from 91% to 100%. The rate of porting from one web browser to the other was seen to be fast without causing changes to the software's features and design.

Under the Educational web application category, Application 3 had the highest score of 99.00% with an acceleration of $0.611 m/s^2$, testing time of 9 sec, speed of 5.5m/s, and transfer rate of 54 while Application 1 was seen to have the lowest score of 92.50% with an acceleration of 0.550 m/s^2, testing time of 10 sec, speed of 5.5 m/s, and transfer rate of 60. Application 2 had a Portability score of 95.625% with an acceleration of 1.0 m/s^2, testing time of 8.5 sec, speed of 8.5 m/s, and transfer rate of 51. Also, Application 4 had 95.00% as the score with an acceleration of 0.906 m/s^2, testing time of 8 sec, speed of 7.25 m/s, and transfer rate of 48. Finally, Application 5 had 96.75% as the score with an acceleration of 0.722 m/s^2, testing time of 9 sec, speed of 6.5 m/s, and transfer rate of 54.

The video and photo applications recorded the highest Portability score of 99.99% and the lowest of 99.00%. Application 9 had the highest score of 99.99% with an acceleration of $0.495 m/s^2$, testing time of 11.5 sec, speed of 5.69 m/s, and transfer rate of 69 while Application 6 was seen to have the lowest score of 99.00% with an acceleration of 0.554

$\mathrm{m/s^2}$, testing time of 11 sec, speed of 6 m/s, and transfer rate of 66. Application 7 had a Portability score of 99.75% with an acceleration of 0.619 $\mathrm{m/s^2}$, testing time of 10.5 sec, speed of 6.5 m/s, and transfer rate of 63. Also, Application 8 had 99.66% as the score with an acceleration of 1.017 $\mathrm{m/s^2}$, testing time of 8.5 sec, speed of 8.65 m/s, and transfer rate of 51. Finally, Application 10 had 99.72% as the score with an acceleration of 2.623 $\mathrm{m/s^2}$, testing time of 6 sec, speed of 15.74 m/s, and transfer rate of 36.

Also, the e-commerce applications recorded the highest Portability score of 99.94% and the lowest of 91.00%. Application 11 had the highest score of 99.94% with an acceleration of 1.570$\mathrm{m/s^2}$, testing time of 6.5 sec, speed of 10.25 m/s, and transfer rate of 39 while Application 12 was seen to have the lowest score of 91.00% with an acceleration of 1.538 $\mathrm{m/s^2}$, testing time of 6.5 sec, speed of 10 m/s, and transfer rate of 39. Application 13 had a Portability score of 98.45% with an acceleration of 1.033 $\mathrm{m/s^2}$, testing time of 7.5 sec, speed of 7.75 m/s, and transfer rate of 45. Also, Application 14 had 95.63% as the score with an acceleration of 0.765 $\mathrm{m/s^2}$, testing time of 8.5 sec, speed of 6.5 m/s, and transfer rate of 51. Finally, Application 15 had 99.00% as the score with an acceleration of 0.778 $\mathrm{m/s^2}$, testing time of 9 sec, speed of 7 m/s, and transfer rate of 54.

The online form creation applications recorded the highest Portability score of 100% and the lowest of 99.75%. The acceleration also ranged from 0.471 $\mathrm{m/s^2}$ to 1.493 $\mathrm{m/s^2}$, testing time ranged from 7.5 sec to 12 sec, speed ranged from 5.65 m/s to 11.20 m/s and transfer rate ranged from 45 to 72.

The company applications recorded the highest Portability score of 99.75% and the lowest of 91.00%. The acceleration also ranged from 0.824 $\mathrm{m/s^2}$ to 1.692 $\mathrm{m/s^2}$, testing time ranged from 7 sec to 9 sec, speed ranged from 7 m/s to 11 m/s and transfer rate ranged from 39 to 54.

Finally, the Document Creation Software recorded the highest Portability score of 100% and the lowest of 99.00%. The acceleration also ranged from 0.822 $\mathrm{m/s^2}$ to 3.40 $\mathrm{m/s^2}$, testing time ranged from 5 sec to 9.7 sec, speed ranged from 8.01 m/s to 17 m/s and transfer rate ranged from 30 to 58.

The average Portability score of Educational, Video editing, E-commerce, Online form creation, Company, and Document Creation Software categories were 95.775%, 99.624%, 96.804%, 99.916%, 96.15%, and 99.66% respectively.

4.4.8 Software Functionality Test

The python script also performed the functionality test by accessing 30 functions on the web application under review. The test was done by clicking on submission forms, live chat feature buttons, social media tabs, internal links, site map aiding in user navigation, print page feature, events calendar, and others to assess if they are functioning as expected. The number of working and non-working functions were recorded for the software functionality evaluation.

Table 4.55 Software Functionality Evaluation

Application Name	Working Functions	Non-working Functions	Score	Functionality (%)
Educational Web Applications				
Application 1	30	0	1	100
Application 2	30	0	1	100
Application 3	30	0	1	100
Application 4	30	0	1	100
Application 5	20	10	0.5	50
Video editing Web applications				
Application 6	30	0	1	100
Application 7	30	0	1	100
Application 8	30	0	1	100
Application 9	30	0	1	100
Application 10	30	0	1	100
E-commerce Software				
Application 11	30	0	1	100
Application 12	30	0	1	100
Application 13	30	0	1	100
Application 14	30	0	1	100
Application 15	30	0	1	100

Table 4.55 Software Functionality Evaluation (cont'd)

Application Name	Working Functions	Non-working Functions	Score	Functionality (%)
Online Form Creation Software				
Application 16	30	0	1	100
Application 17	30	0	1	100
Application 18	30	0	1	100
Application 19	30	0	1	100
Application 20	30	0	1	100
Company Web applications				
Application 21	30	0	1	100
Application 22	30	0	1	100
Application 23	30	0	1	100
Application 24	30	0	1	100
Application 25	28	2	0.92	92
Document Creation Software				
Application 26	30	0	1	100
Application 27	30	0	1	100
Application 28	30	0	1	100

According to Table 4.55, web applications under all categories performed the functionality test with thirty (30) working functions except Application 5 in the Educational web application category, which recorded twenty (20) functioning features with ten (10) non-functioning features. Application 5 was seen to have some non-functioning tabs, some non-functioning internal tabs, and other functional problems. Application 25 in the Company web application category also had twenty-eight (28) functioning features with two (2) non-functioning features. The other web applications had a Functionality score of 100% each.

The average Functionality score of Educational, Video editing, E-commerce, Online form creation, Company, and Document Creation software categories were 90%, 100%, 100%, 100%, 98.40%, and 100%, respectively.

4.4.9 Software Availability Test

The quality assurance model evaluated the web applications for availability by calculating the total downtime, the number of operational hours, and the number of failures to find the MTBF and MTTF for Availability evaluation. The python script was automated to send HTTP requests to the website being evaluated at every 10 seconds for a period of 3600 seconds. The time the web applications were unavailable for usage was recorded as the total downtime This is as shown in Table 4.56.

Table 4.56 Software Availability Evaluation

Application Name	Total Downtime	Operational Time	Number of Failures	Score (%)
Educational Web Applications				
Application 1	0.010% or 1 min 20 secs	3520 sec	8	97.93
Application 2	0.031% or 5 min 4 secs	3296 sec	27	91.29
Application 3	0.013% or 1 min 59 secs	3481 sec	10	97.61
Application 4	0.016% or 2 min 45 secs	3435 sec	13	97.38
Application 5	0.043% or 5 min 36 sec	3264 sec	32	90.87
Video editing Web applications				
Application 6	0.004% or 25 secs	3575 sec	2	99.12
Application 7	0% or 0 min 5 sec	3595 sec	1	100
Application 8	0% or 0 min 5 sec	3595 sec	1	100
Application 9	0.005% or 40 secs	3560 sec	3	99.03
Application 10	0% or 0 min 5 sec	3595 sec	1	100
E-commerce Software				
Application 11	0.019% or 2 min 15 secs	3465 sec	15	96.99
Application 12	0% or 0 min 5 sec	3595 sec	1	100
Application 13	0% or 0 min 5 sec	3595 sec	1	100
Application 14	0.014% or 2 min 5 secs	3475 sec	11	97.54
Application 15	0% or 0 min 5 sec	3595 sec	1	100

Table 4.56 Software Availability Evaluation (cont'd)

Application Name	Total Downtime	Operational Time	Number of Failures	Score (%)
Online Form Creation Software				
Application 16	0% or 0 min 5 sec	3595 sec	1	100
Application 17	0% or 0 min 5 sec	3595 sec	1	100
Application 18	0% or 0 min 5 sec	3595 sec	1	100
Application 19	0% or 0 min 5 sec	3595 sec	1	100
Application 20	0% or 0 min 5 sec	3595 sec	1	100
Company Web applications				
Application 21	0.006% or 1 min 5 secs	3535 sec	5	98.35
Application 22	0.021% or 2 min 48 secs	3432 sec	17	96.75
Application 23	0.013% or 1 min 59 secs	3481 sec	10	97.61
Application 24	0.010% or 1 min 20 secs	3520 sec	8	97.93
Application 25	0.029% or 5 min 2 secs	3298 sec	25	91.45
Document Creation Software				
Application 26	0% or 0 min 5 sec	3595 sec	1	100
Application 27	0% or 0 min 5 sec	3595 sec	1	100
Application 28	0% or 0 min 5 sec	3595 sec	1	100

Table 4.56 shows that the applications had been available for use more than 90% of the time. Web applications under the Educational web application category recorded eight (8) failures, twenty-seven (27) failures, ten (10) failures, thirteen (13) failures, and thirty-two (32) failures for Applications 1, 2, 3, 4, and 5, respectively. Application 1 recorded a downtime of 1 minute 20 seconds, Application 2 recorded a downtime of 5 minutes 4 seconds, Application 3 recorded a downtime of 1 minute 59 seconds, Application 4 recorded a downtime of 2 minutes 45 seconds, and Application 5 recorded a downtime of 5 minutes 36 seconds. Applications 5 and 2 experienced longer downtime than the other Educational web applications. The average Availability score was seen to be 95.016%.

Applications in the Video editing category also recorded two (2) failures, one (1) failure, one (1) failure, three (3) failures, and one (1) failure for Applications 6, 7, 8, 9, and 10, respectively. Application 6 recorded a downtime of 25 seconds, Application 7 recorded a downtime of 5 seconds, Application 8 recorded a downtime of 5 seconds, Application 9 recorded a downtime of 40 seconds and Application 10 recorded 5 seconds. The average Availability score was seen to be 99.63%.

Three (3) applications under the e-commerce category experienced downtimes of 5 seconds each with one (1) failure and were seen to have a higher Availability score of 100%. Application 11 had a downtime of 2 minutes 15 seconds with 15 failures whiles Application 14 had a downtime of 2 minutes 5 seconds with 11 failures.

Under the online form creation software category, all the applications performed well in the test with downtimes of 5 seconds each. They all had 1 failure and recorded 100% as the Availability score.

Application 21 recorded a downtime of 1 minute 5 seconds, Application 22 recorded 2 minutes 48 seconds, Application 23 recorded 1 minute 59 seconds, Application 24 recorded 1 minute 20 seconds while Application 25 recorded 5 minutes 2 seconds under the company web applications category. Application 21 was seen to have performed better than all applications in the category and was seen to have been more available with an operational time of 3535 seconds.

All applications under the Document Creation Software category performed well with 5 seconds of downtime and one (1) failure.

4.4.10 Software Reusability Test

The reusability rate of the web applications is shown in Table 4.57. The software delivery time was calculated from the operational hours and downtime to get the overall reusability rate.

Table 4.57 Software Reusability Evaluation

Application Name	Operational Time	Total Downtime	Reusability Score	Reusability Scale (%)
Educational Web Applications				
Application 1	3520 sec	0.010% or 1 min 20 secs	1.4015	96.77
Application 2	3296 sec	0.031% or 5 min 4 secs	1.2887	89.47
Application 3	3481 sec	0.013% or 1 min 59 secs	1.3939	95.15
Application 4	3435 sec	0.016% or 2 min 45 secs	1.3184	94.28
Application 5	3264 sec	0.043% or 5 min 36 sec	1.1265	85.03
Video editing Web applications				
Application 6	3575 sec	0.004% or 25 secs	1.3852	90.04
Application 7	3595 sec	0% or 0 min 5 sec	1.5184	100
Application 8	3595 sec	0% or 0 min 5 sec	1.5184	100
Application 9	3560 sec	0.005% or 40 secs	1.3895	90.18
Application 10	3595 sec	0% or 0 min 5 sec	1.5184	100
E-commerce Software				
Application 11	3465 sec	0.019% or 2 min 15 secs	1.1802	82.98
Application 12	3595 sec	0% or 0 min 5 sec	1.5184	100
Application 13	3595 sec	0% or 0 min 5 sec	1.5184	100
Application 14	3475 sec	0.014% or 2 min 5 secs	1.207	85.73
Application 15	3595 sec	0% or 0 min 5 sec	1.5184	100
Online Form Creation Software				
Application 16	3595 sec	0% or 0 min 5 sec	1.5184	100
Application 17	3595 sec	0% or 0 min 5 sec	1.5184	100
Application 18	3595 sec	0% or 0 min 5 sec	1.5184	100
Application 19	3595 sec	0% or 0 min 5 sec	1.5184	100
Application 20	3595 sec	0% or 0 min 5 sec	1.5184	100
Company Web applications				
Application 21	3535 sec	0.006% or 1 min 5 secs	1.5184	100
Application 22	3432 sec	0.021% or 2 min 48 secs	1.5184	100
Application 23	3481 sec	0.013% or 1 min 59 secs	1.3939	92.07
Application 24	3520 sec	0.010% or 1 min 20 secs	1.5184	100
Application 25	3298 sec	0.029% or 5 min 2 secs	1.0265	79.54

Table 4.57 Software Reusability Evaluation (cont'd)

Application Name	**Operational Time**	**Total Downtime**	**Reusability Score**	**Reusability Scale (%)**
Document Creation Software				
Application 26	3595 sec	0% or 0 min 5 sec	1.5184	100
Application 27	3595 sec	0% or 0 min 5 sec	1.5184	100
Application 28	3595 sec	0% or 0 min 5 sec	1.5184	100

According to Table 4.57, web applications under the Educational web application category had a Reusability scale ranging from 85% to 97% with Application 5 scoring the lowest value and Application 1 scoring the highest value. Application 3 had the second-highest score of 95.15% in the category. The average Reusability scale was 92.14%.

The Video editing applications also had three (3) applications scoring 100% with an average Reusability scale of 96.044%. Application 6 had a downtime of 25 seconds and a Reusability score of 1.3852 with a percentage of 90.04. Application 9 also had a downtime of 40 seconds and a Reusability score of 1.3895 with 90.18% as the Reusability scale.

The e-commerce software also had an average Reusability scale of 93.742% with Applications 12, 13, and 15 having 100% as the Reusability scale. Application 11 was seen to have a downtime of 2 minutes 15 seconds with a Reusability scale of 82.98%. Application 14 had a downtime of 2 minutes 5 seconds and a Reusability scale of 85.73%.

All applications in the form creation software category had a Reusability scale of 100% with downtimes of 5 seconds each and operational time 3595 seconds. The components of the software were seen to be reusable. There was an average of 100% as the Reusability score.

The company web applications also had a Reusability score ranging from 100% to 79%. Application 21, Application 22, and Application 24 had 100% reusable components; Application 23 had 92.07% as the Reusability score with downtime of 1 minute 59 seconds while Application 25 had 79.54% of reusable components. The average was an average Reusability score of 94.322%.

The Document Creation Software also had an average Reusability score of 100% with a downtime of 5 seconds.

4.4.11 Software Security Test

A security test was performed by the quality assurance software to assess the vulnerability of the web applications and is as shown in table 4.58.

In testing for a vulnerability like injection (A1), the written python script looked for input fields and URL parameters in the web application under review. It then sent an input containing a single quotation mark, eg. id = ‘1’. Then, based on the response from the web server, the application determined if it is vulnerable or not by checking to see if the response contains SQL errors. The presence of these SQL errors makes it vulnerable to injection.

Also, in testing for Cross site request forgery (A8) attack, the session management of the software was checked to see if it is vulnerable. This was done by checking if session management relies only on client-side values, if so, then the web application was seen to be vulnerable.

Table 4.58 Security Test Results

Application Name	**A_1**	**A_2**	**A_3**	**A_4**	**A_5**	**A_6**	**A_7**	**A_8**	**A_9**	**A_{10}**	**Security Score (%)**
Educational Web Applications											
Application 1	10	10	10	10	10	10	10	0	10	10	90
Application 2	10	10	10	10	10	10	10	0	10	10	90
Application 3	10	10	10	10	10	10	10	10	10	10	100
Application 4	10	10	10	10	10	10	10	10	10	10	100
Application 5	10	10	10	10	10	10	10	0	10	10	90
Video editing Web applications											
Application 6	10	10	10	10	10	10	10	10	10	10	100
Application 7	10	10	10	10	10	10	10	10	10	10	100
Application 8	10	10	10	10	10	10	10	10	10	10	100
Application 9	10	10	10	10	10	10	10	10	10	10	100
Application 10	10	10	10	10	10	10	10	10	10	10	100

Table 4.58 Security Test Results (cont'd)

Application Name	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	Security Score (%)
E-commerce Software											
Application 11	10	10	10	10	10	10	10	10	10	10	100
Application 12	10	10	10	10	10	10	10	10	10	10	100
Application 13	10	10	10	10	10	10	10	10	10	10	100
Application 14	10	10	10	10	10	10	10	10	10	10	100
Application 15	10	10	10	10	10	10	10	10	10	10	100
Online Form Creation Software											
Application 16	10	10	10	10	10	10	10	10	10	10	100
Application 17	10	10	10	10	10	10	10	10	10	10	100
Application 18	10	10	10	10	10	10	10	10	10	10	100
Application 19	10	10	10	10	10	10	10	10	10	10	100
Application 20	10	10	10	10	10	10	10	10	10	10	100
Company Web applications											
Application 21	10	10	10	10	10	10	10	0	10	10	90
Application 22	10	10	10	10	10	10	10	0	10	10	90
Application 23	10	10	10	10	10	10	10	0	10	10	90
Application 24	10	10	10	10	10	10	10	0	10	10	90
Application 25	10	10	10	10	10	10	10	0	10	10	90
Document Creation Software											
Application 26	10	10	10	10	10	10	10	10	10	10	100
Application 27	10	10	10	10	10	10	10	10	10	10	100
Application 28	10	10	10	10	10	10	10	10	10	10	100

Results from Table 4.58 show that three applications under the Educational web application category (Applications 1, 2, and 5) scored 90% while two other applications (Application 3 and Application 4) scored 100%. The applications that scored 90% failed the vulnerability level A_8 which represents the Cross-Site Request Forgery attack.

All applications under the company web application category also failed the vulnerability level A_8 and hence had a security score of 90%.

Applications under the other categories passed all the security vulnerability tests with an overall score of 100%.

4.4.12 Software Cost Estimate Test

The web applications were grouped under each of the existing categories of web applications namely retail, financial, news and information portals, and entertainment in order to get the range of cost estimate. The grouping was done using a survey method that was submitted to ten (10) experts in software development. The grouping was done as follows: e-commerce was grouped under retail; educational, company, Document Creation Software and online form software were grouped under news and information portals and Video editing was grouped under entertainment. The cost estimate of the web applications was calculated as shown in table 4.59. The line of codes, function points, labour, and effort were employed to find the software cost estimate.

Table 4.59 Cost Evaluation based on software category

Application Name	FP	KLOC	Effort	Labour	Software Cost Estimate ($)	Score
Educational Web Applications						
Application 1	457.96	23.81	71.35	500	35,677.16	100
Application 2	478.29	24.87	74.74	500	37,374.42	100
Application 3	474.01	24.64	74.03	500	37,016.68	100
Application 4	462.24	24.03	72.06	500	36,034.34	100
Application 5	435.49	22.64	67.61	500	33,807.38	100
Video editing Web applications						
Application 6	550.80	31.94	97.71	500	48,855.41	100
Application 7	533.25	30.92	94.38	500	47,191.66	100
Application 8	526.50	30.54	93.10	500	46,552.76	100

Table 4.59 Cost Evaluation based on software category (cont'd)

Application Name	FP	KLOC	Effort	Labour	Software Cost Estimate ($)	Score
Video editing Web applications						
Application 9	531.90	30.85	94.13	500	47,063.83	100
Application 10	521.10	30.22	92.08	500	46,042.06	100
E-commerce Software						
Application 11	507.18	26.88	81.22	500	40,614.39	100
Application 12	518.95	27.50	83.24	500	41,623.70	100
Application 13	509.32	26.99	81.59	500	40,797.78	100
Application 14	513.60	27.22	82.32	500	41,164.72	100
Application 15	525.37	27.84	84.34	500	42,174.92	100
Online Form Creation Software						
Application 16	531.90	30.85	94.13	500	47,063.83	100
Application 17	525.15	30.45	92.85	500	46,425.05	100
Application 18	518.40	30.07	91.57	500	45,786.85	100
Application 19	538.65	31.24	95.40	500	47,703.18	100
Application 20	544.05	31.55	96.43	500	48,215.06	100
Company Web applications						
Application 21	440.84	22.92	68.50	500	34,251.96	100
Application 22	408.78	21.25	63.18	500	31,590.26	100
Application 23	436.56	22.70	67.79	500	33,896.26	100
Application 24	426.93	22.20	66.19	500	33,096.00	100
Application 25	391.62	20.36	60.35	500	30,176.59	100
Document Creation Software						
Application 26	564.30	32.73	100.26	500	50,137.76	100
Application 27	550.80	31.95	97.71	500	48,855.41	100
Application 28	561.60	32.57	99.76	500	49,881.12	100

The table shows that all the web applications under the news and information portals category had a 100% score for software cost estimate. The 100% score was achieved because the cost estimate of news and information portals range between $2,500 and $600,000 and none of the applications exceeded $600,000.

Also, the web applications under the retail web applications category had a score of 100% because they did not exceed $210,000.

The web applications under the entertainment web applications category had a score of **100%** because they ranged between $40,000 and $100,000.

4.5 Voting Method

The voting method was carried out by multiplying the criteria weights from the AHP technique with the scores generated by each quality attribute in the software quality assurance model. These are summed up at the summing junction and displayed to the user as a percentage of software quality. The overall score from the voting method varies between 0 and 100% and shows the percentage of software quality for each evaluated web application. The higher the score from the software quality evaluation, the higher quality of the web application. Results for the voting method are shown in Figures 4.10 to 4.15.

The results from the voting model show scores for each of the eleven (11) software quality attributes that have been extended to twenty-four (24) since there are thirteen (13) sub-attributes in the proposed software quality assurance model.

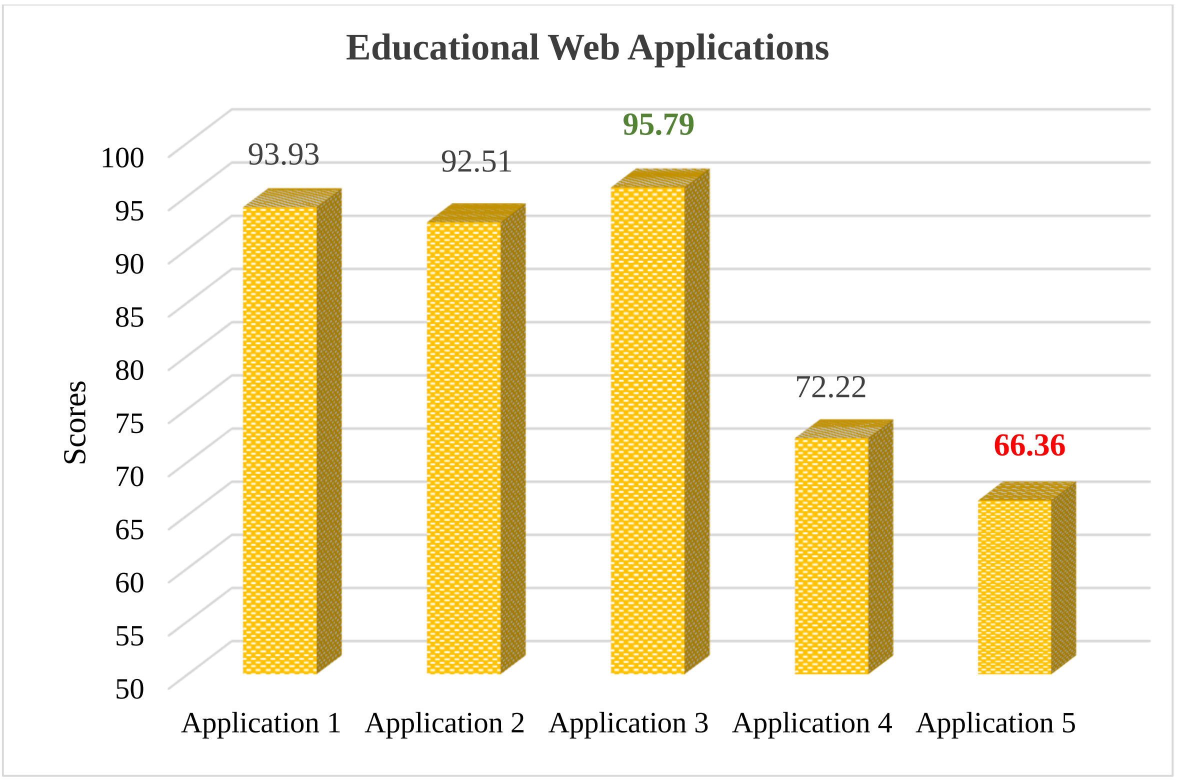


Figure 4.10 Results for Educational Web Applications

Figure 4.10 shows the results from the voting model for each of the educational web applications. Application 3 is seen to have the highest quality score of 95.79% while Application 5 is seen to have the lowest score of 66.36%. Applications 1, 2, and 3 had software quality scores that are greater than 90% while the remaining two (2) applications had lesser scores.

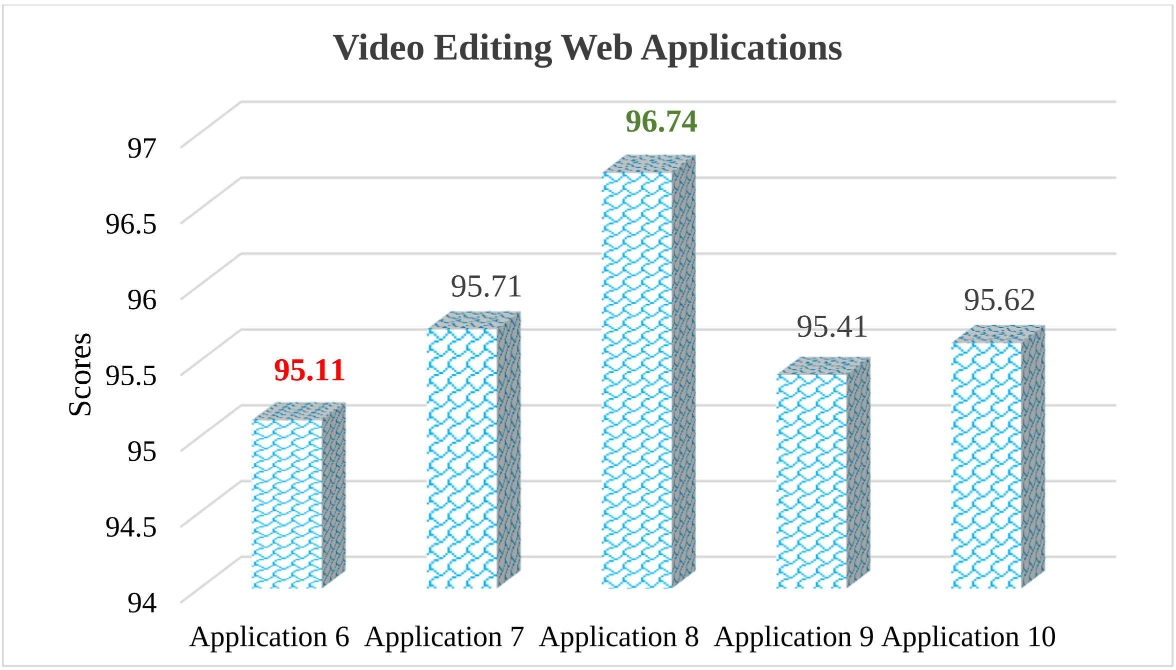


Figure 4.11 Results for Video editing Web Applications

It can be seen from Figure 4.11 that Application 8 had the highest software quality score of 96.74% from the voting model while Application 6 had the lowest score with 95.11%. All the applications were seen to have a score greater than 95% and can be said to have performed well.

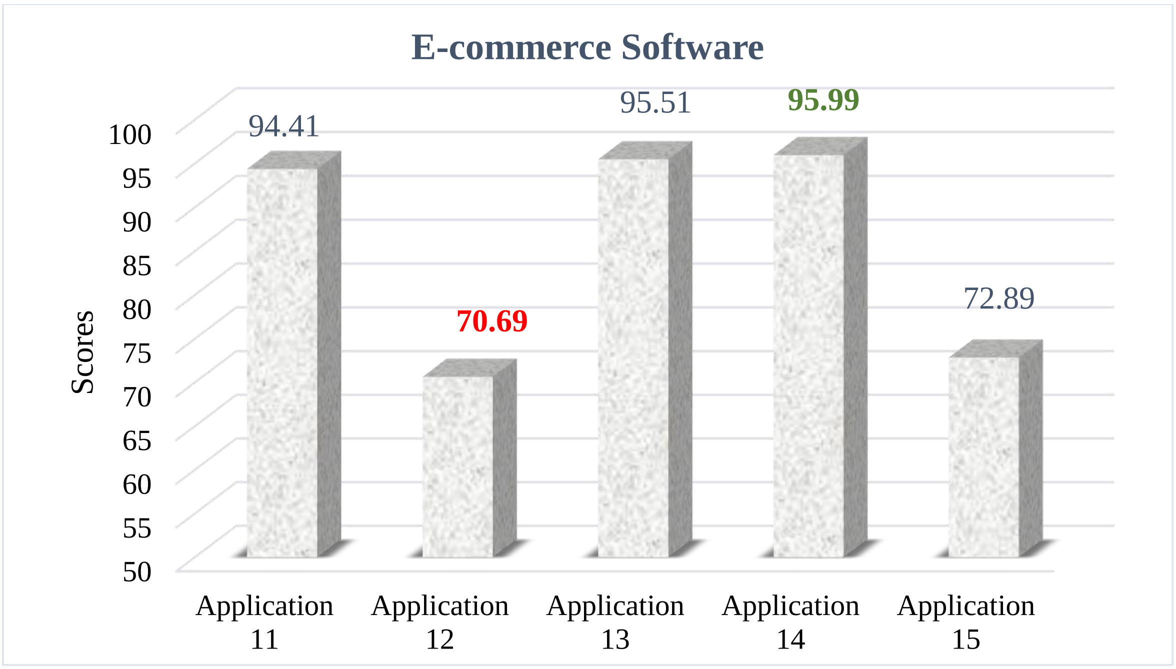


Figure 4.12 Results for E-commerce Software

It is illustrated in Figure 4.12 that quality scores for the e-commerce software are 94.41%, 70.69%, 95.51%, 95.99%, and 72.89% for Applications 11, 12, 13, 14, and 15, respectively. Application 14 had the highest score while Application 12 had the lowest score.

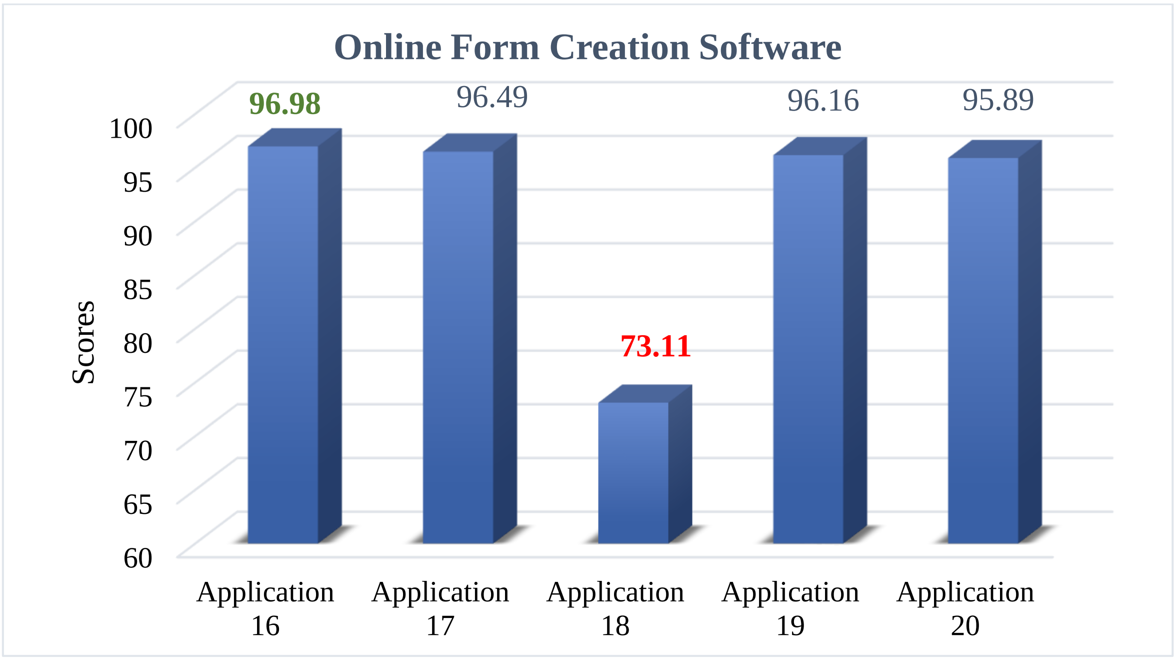


Figure 4.13 Results for Online Form creation Software

Figure 4.13 shows a graphical view of the results for online form creation software after being applied to the voting model. Application 16 had the highest score of 96.98% while Application 18 had the lowest score with 73.11%. Four (4) applications had quality scores greater than 95% while the score of one (1) of the applications was less than 95%.

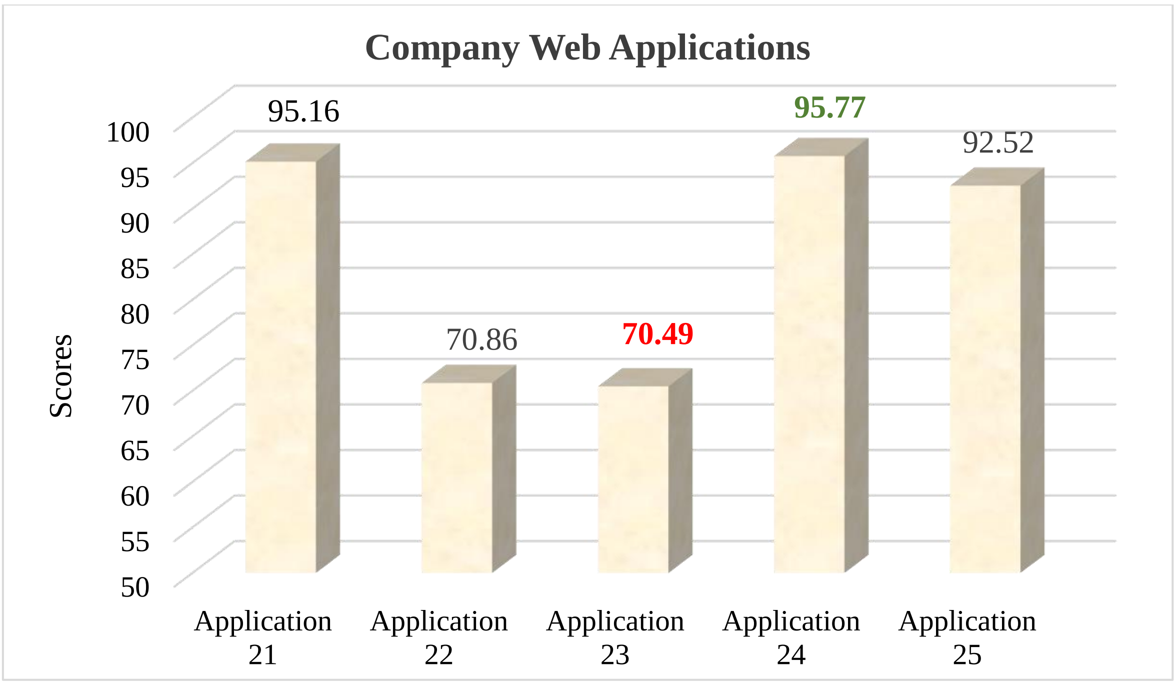


Figure 4.14 Results for Company Web Applications

It is illustrated in Figure 4.14 that two (2) of the company web applications had software quality scores of 70.86% and 70.49% while the remaining three (3) had scores above 90%. Application 24 had the highest score of 95.77% while Application 23 had the lowest score of 70.49%.

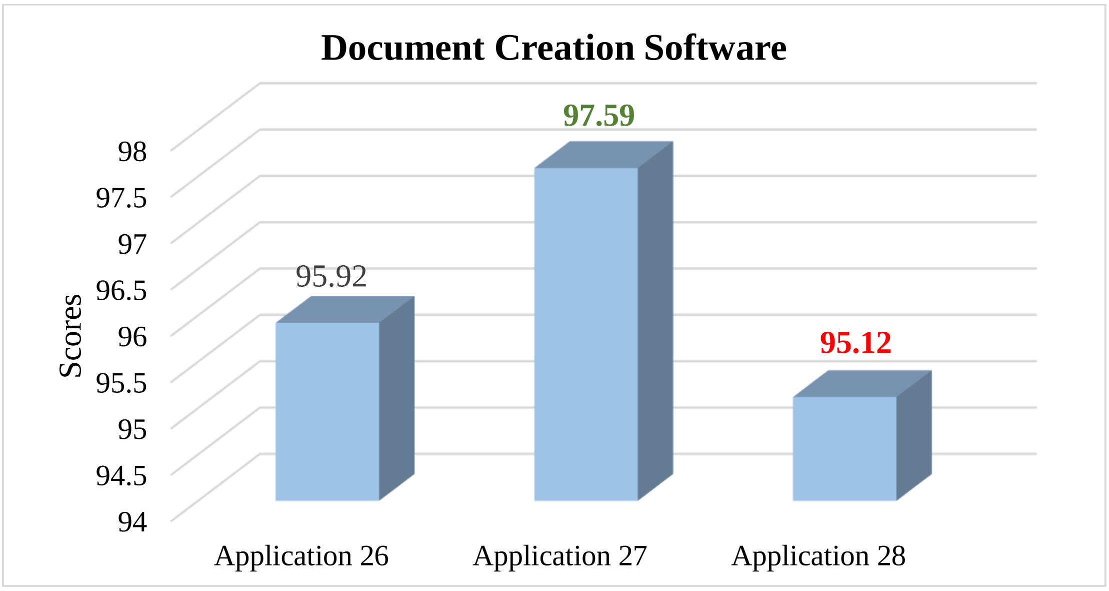


Figure 4.15 Results for Document Creation Software

It is seen in Figure 4.15 that Application 27 had the highest score of 97.59% in the category while Application 28 had the lowest score with 95.12%.

The average results from the voting model for each software category is shown in Figure 4.16.

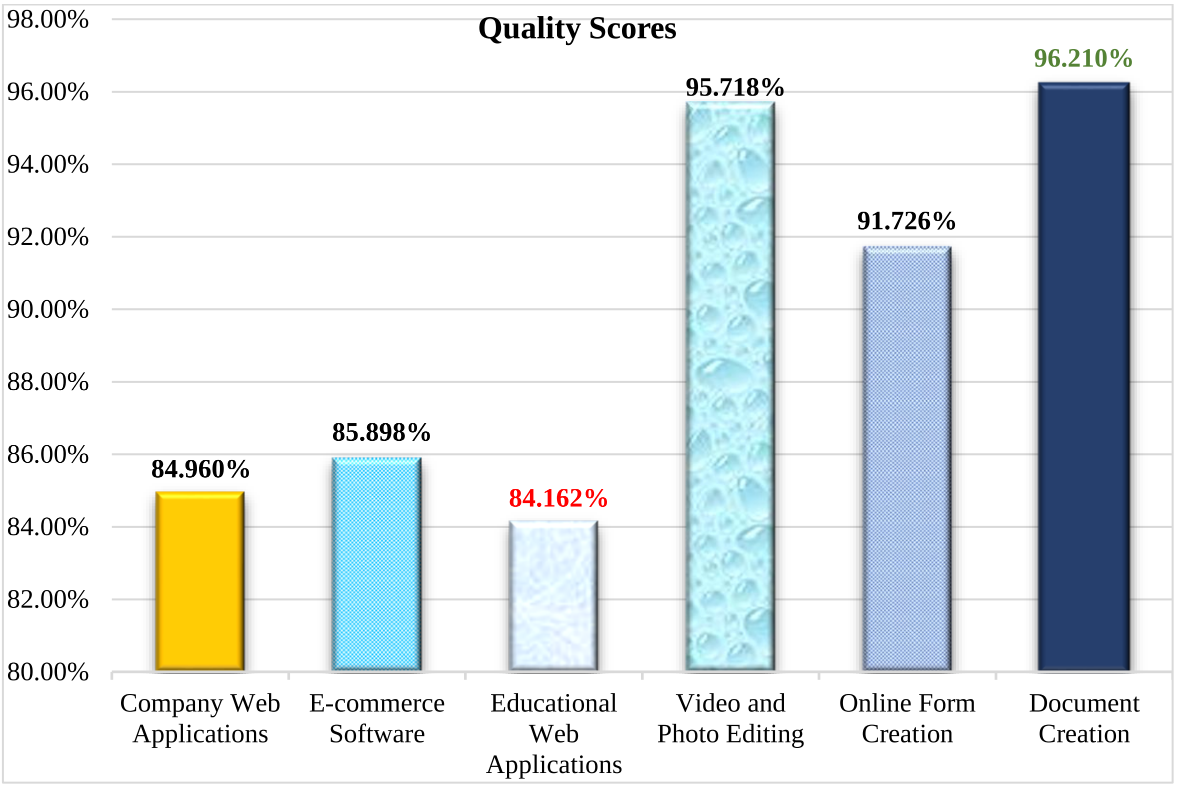


Figure 4.16 Average Result of Web-Applications

Figure 4.16 shows that the category with the highest score is Document Creation Software while the category with the lowest score is Educational web applications. It was seen from the results that most of the web applications failed various tests due to numerous reasons such as server downtime or breakdown, bad programming practices or coding issues, network problems, among others.

When the selected Educational web applications were assessed for Reliability, some of the web applications had failure rates of 120 and 130. This showed that the applications had poor error handling capabilities when subjected to extremely heavy conditions. On assessing the applications for Usability based on a survey approach, users gave higher scores to the Document Creation applications because the applications were easy to navigate, had clearer organisation of information, had lesser load time among others. Also, some of the Educational web applications had non-working functions when assessed for functionality.

This resulted in the overall average score from the voting model leading to the low score for Educational web applications and high score for the Document Creation Software.

4.6 Performance Evaluation

To evaluate the performance of the proposed software quality assurance model, the research works by Bayu and Banowosari (2021), Kaur, Kaur and Kaur (2016) and Budiman *et al*. (2018) were used. The three works were selected due to their recency and the use of standard software quality models and attributes which were equally applied throughout this thesis.

Bayu and Banowosari (2021) used the attributes of the ISO 9126 model to propose a model to evaluate the application of "PT Karya Prima Usahatama" with the URL of http://sip.kpusahatama.id/. In Kaur, Kaur and Kaur (2016), the standard quality attribute, Efficiency, was used to evaluate Punjab University, Chandigarh with the URL of http://puchd.ac.in/. Also, Budiman *et al*. (2018) used Efficiency, Reliability, and Portability to evaluate the performance of a student academic portal with the URL of https://sia-dev.unmul.ac.id/. The proposed software quality assurance model was used to evaluate the works done by the three authors and is represented in Tables 4.61, 4.62, and 4.63.

Table 4.61 Comparison of Proposed Model with Bayu and Banowosari (2021)

Software Quality Attribute	**Bayu and Banowosari (2021)**	**Proposed Quality Assurance Model**
Functionality	100%	100%
Usability	85%	92%
Efficiency	74.5%	93%
Reliability	100%	100%
Portability	100%	100%
Maintainability	100%	100%
Security	-	90%
Testability	-	100%
Reusability	-	85.79%
Availability	-	95.78%
Cost	-	100%
Average of Score	**93.25%**	**96.05%**

Table 4.61 shows an illustration of a comparison done between the model of Bayu and Banowosari (2021) and the proposed software quality assurance model. The proposed model outperformed the results from Bayu and Banowosari (2021) during evaluation. The software named kpusahatama with the URL of http://sip.kpusahatama.id/ had a score of 100% for Functionality in the proposed model and had 100% also in Bayu's evaluation. It scored 92% for Usability in the proposed model and had 85% in Bayu's evaluation. Efficiency's score was 93% in the proposed model and 74.5% in Bayu's evaluation. Reliability, Portability and Maintainability scores were 100% in both evaluations. In the proposed model, Security, Testability, Reusability, Availability, and Cost had scores of 90%, 100%, 85.79%, 95.78%, and 100%, respectively. The average of the scores for the proposed model with six (6) quality attributes comprising of Functionality, Usability, Efficiency, Reliability, Portability, and Maintainability was 97.50% and that of Bayu's evaluation with six attributes was 93.25%. When extended to eleven (11) attributes, the proposed model had an average score of 96.05%.

Table 4.62 Comparison of Proposed Model with Kaur, Kaur and Kaur (2016)

Software Quality Attribute	**Kaur, Kaur and Kaur (2016)**	**Proposed Quality Assurance Model**
Efficiency	85%	95.298%
Functionality	-	100%
Usability	-	85.10%
Availability	-	92%
Reliability	-	100%
Portability	-	99.75%
Maintainability	-	99.48%
Security	-	90%
Testability	-	100%
Reusability	-	98.80%
Cost	-	100%
Average Score	**85%**	**96.40%**

Table 4.62 illustrates the comparison made between the proposed software quality assurance model and Kaur, Kaur and Kaur (2016). Kaur's work evaluated the efficiency of Punjab University, Chandigarh with the URL of http://puchd.ac.in/ and had a percentage score of 94%. The same software was evaluated with the proposed model for software quality attributes such as Availability, Functionality, Usability, Efficiency, Reliability, Portability, Maintainability, Security, Testability, Reusability, and Cost with scores of 95.298%, 100%, 85.10%, 92%, 100%, 99.75%, 99.48%, 90%, 100%, 98.80%, and 100%, respectively. The proposed model had an average score of 96.40% when extended to eleven (11) attributes while Kaur model had 85% for efficiency evaluation.

Table 4.63 Comparison of Proposed Model with Budiman *et al*. (2018)

Software Quality Attribute	**Budiman *et al*. (2018)**	**Proposed Quality Assurance Model**
Efficiency	66.5%	85%
Reliability	100%	100%
Portability	100%	100%
Usability	0	89.37%
Maintainability	-	99.909%
Functionality	-	100%
Reusability	-	98.78%
Security	-	90%
Cost	-	100%
Availability	-	97.259%
Testability	-	100%
Average	**88.83%**	**96.39%**

Table 4.63 shows a comparison of an evaluation on “unmul software” done by Budiman *et al*., (2018) with the URL of https://sia-dev.unmul.ac.id/ and the proposed software quality assurance model. The proposed model is seen to have an average score of 96.39% when the software was evaluated against the eleven (11) software quality attributes and 95% when evaluated against three (3) attributes by Budiman’s model. There was a score of 66.5% for Efficiency, 100% for Reliability and 100% for Portability in Budiman’s model while there was 85% for Efficiency, 100% for Reliability, 100% for Portability, 89.37% for Usability, 99.909% for Maintainability, 100% for Functionality, 98.78% for Reusability, 90% for Security, 100% for Software Cost Estimate, 97.259% for Availability and 100% for Testability.

CHAPTER 5
CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The research reviewed standard and well-known software quality models and identified the various attributes, leading to the achievement of the first objective. These attributes were grouped under eleven (11) main attributes and thirteen (13) sub-attributes. Again, a multi-criteria decision-making analysis of the main software quality attributes was carried out using AHP to achieve the second objective. Results from the AHP evaluation ranked maintainability with the highest score of 17.37% and cost with the lowest score of 4.73%. This resulted in the development of a model for assessing the quality of software factoring in the major attributes of software quality assurance. The developed model will assist software developers and end-users greatly in developing and assessing software quality. The research finally, evaluated the developed model using standard metrics.

The focus of this research was to develop a model to assess the quality of software, most importantly, software used in safety-critical parts of organisations since the use of less quality software in such organisations could lead to adverse effects. Moreover, clients expect quality software to be developed for them, hence, the need for an evaluation mechanism. It, therefore, became imperative to satisfy user’s needs by developing a model to aid them in evaluating the quality of developed software. Web-based applications were targeted in the research since they can be accessed by users over a network such as an internet or intranet anywhere and at every time of the day.

The developed model was used to evaluate the quality of twenty-eight (28) web applications grouped under six (6) categories: Educational, E-commerce, Company, Online Document Creation Software, Video editing, and Form creation web applications. Results from the evaluation showed that Document Creation software had the highest average quality score of 96.21% while Educational web applications had the lowest average score of 84.16%.

The quality assurance model was developed using mathematical models of the eleven (11) main attributes. The scores from the mathematical models were brought together with the use of a voting model which assigns the criteria weights obtained from AHP evaluation to the output in order to get the overall score for software quality. Furthermore, an access control model comprising of BLP and Biba model was designed to secure the quality assurance model to regulate who can access it. This was evaluated for accuracy, precision, recall and F1 score and had values of 0.93, 0.96, 0.91, and 0.92, respectively.

The performance of the quality assurance model was evaluated using research works by Bayu and Banowosari (2021), Kaur, Kaur and Kaur (2016), and Budiman *et al*., (2018) were used. The three (3) studies were selected due to their recency and the use of standard software quality models and attributes. The performance evaluation showed that the proposed model outperformed the selected models when evaluated against the attributes they used and when extended to the use of eleven (11) quality attributes.

5.2 Recommendations

Despite the performance of the quality assurance model, some web applications performed poor when evaluated for some attributes. This poor performance could be due to numerous reasons; hence, it is also recommended that future works investigate other reasons why failure of some of the quality tests occurred.

Results from AHP showed that based on experts' judgement, some attributes were ranked and assigned higher weights. This may likely be due to the nine (9) point scale of comparison employed by AHP. It is recommended that other methods of assigning weights to the attributes be considered. Also, future works may consider assigning weights to all twenty-four (24) quality attributes to determine the maximum weights of the attributes.

REFERENCES

Abrahamyan, S., Balyan, S., Muradov, A., Korkhov, V., Moskvicheva, A. and Jakushkin, O. (2016), "Development of M-Health Software for People with Disabilities", *International Conference on Computational Science and Its Applications*, pp 468 - 479.

Alanazi, T. S., Akour, M., Anbar, M. and Alsadoun, A. (2019), "Enterprise Resource Planning Quality Model ERPQM", *First International Conference of Intelligent Computing and Engineering (ICOICE), IEEE*, pp. 1–5.

Al-Badareen, A. B., Selamat, M. H., Jabar, M. A., Din, J. and Turaev, S. (2011), "Software Quality Models: A Comparative Study", *International Conference on Software Engineering and Computer Systems (ICSECS)*, Vol. 179, pp. 46–55.

Aldabbas, M. and Teufel, B. (2016), "Human Aspects of Smart Technologies' Security: The Role of Human Failure", *Journal of Electronic Science and Technology*, Vol. 14, No. 4, pp. 311 - 318.

Alese, B. K., Olojo, O. J., Adewale, O. S., Adetunmbi, A. A. and Falaki, S. O. (2007), "Factor Analytic Approach to Computer Network/ Information Security Awareness in South-Western Nigeria", *Pacific Journal of Science and Technology*, Vol. 8, No. 2, pp.351 -366.

Ali, N., Daneth, H. and Hong, J. E. (2020), "A hybrid DevOps process supporting software reuse: A pilot project", *Journal of Software: Evolution and Process*, Vol. 32, No. 7, 16 pp.

Aliu, F., Ayeni, O. A., Thompson, A. F. and Alese, B. K. (2020), "Information Security Risk Analysis Using Analytic Hierarchy Process and Fuzzy Comprehensive Evaluation", *International Journal of Computer Science and Information Security*, Vol. 18, No. 6, pp. 36 - 45.

Al-Khurafi, O. B. and Al-Ahmad, M. A. (2015), "Survey of Web Application Vulnerability Attacks", *4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, pp. 154-158

Al-Nawaiseh, J. A., Helmy, Y. and Khalil, E. (2020), "A New Software Quality Model for Academic Information Systems: Case Study E-Learning System", *International Journal of Scientific and Technology Research*, Vol. 9, No. 01, pp. 271–282.

Al-Obaithani, F. S. and Ameen, A. A. (2018), "Towards Customised Smart Government Quality Model", *International Journal of Software Engineering and Applications (IJSEA)*, Vol. 9, No. 2, pp. 41 – 51.

Al-Qutaish, R. E. (2010), "Quality Models in Software Engineering Literature: An Analytical and Comparative Study", *Journal of American Science*, Vol. 6, No. 2, pp. 166-175.

Anon., (2001), "Software Engineering - Product Quality - Part 1: Quality Model", *ISO/IEC JTC 1/SC 7 Software and systems engineering*, Vol. 1, 25 pp.

Anon., (2021a), "Difference between System Software and Application Software", https://www.guru99.com/difference-system-software-application-software.html, Accessed: May 25, 2021.

Anon., (2021b), "System Software", https://techterms.com/definition/systemsoftware, Accessed: May 20, 2021.

Anon., (2021c), https://www.techopedia.com/definition/2953/mobile-application-mobile-app, Accessed: May 20, 2021.

Anon., (2021d), "Web-Based Application: What It Is, and Why You Should Use It", https://lvivity.com/web-based-applications, Accessed: May 20, 2021.

Anon., (2021e), "How Much Should a Website Cost in 2021?", https://www.webfx.com/How-much-should-web-site-cost.html, Accessed: March 20, 2021.

Anon., (2021f), "OWASP Top 10 Security Risks and Vulnerabilities", https://sucuri.net/guides/owasp-top-10-security-vulnerabilities-2021/, Accessed: May 13, 2021.

Aull-Hyde, R. and Davis, K. A. (2012), "Military applications of the analytic hierarchy process", *International Journal of Multicriteria Decision Making,* Vol. 2, No. 3, 7 pp.

Ayachi, Y., Ettifouri, E. H., Berrich, J. and Toumi, B. (2019), "Modeling the OWASP Most Critical WEB Attacks", In *Information Systems and Technologies to Support Learning,* Rocha, Á. and Serrhini, M. (eds), *Proceedings of Smart Innovation, Systems and Technologies,* Vol 111. Springer, Cham, 420pp.

Balaji, N., Shivakumar, N. and Ananth, V. V. (2013), "Software Cost Estimation using Function Point with Non-Algorithmic Approach", *Global Journal of Computer Science and Technology, Software and Data Engineering,* Vol. 13, No. 8, 7 pp.

Balamurugan, B., Gnana, S. N., Monisha V. and Saranya, V. (2015), "A Honey Bee Behavior Inspired Novel Attribute - Based Access Control using Enhanced Bell-Lapadula Model in Cloud Computing", *International Conference on Innovation Information in Computing Technologies (ICIICT), Chennai, India,* pp. 1 – 6.

Bayu, F. and Banowosari, L. Y. (2021), "Quality Analysis of Payroll Information System Based on ISO 9126 In PT Karya Prima Usahatama*", International Journal of Research Publications,* Vol. 71, No. 1, 11 pp.

Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., and Merritt, M. (1978) *Characteristics of Software Quality,* North Holland, 150 pp.

Buckleton, J. S., Curran, J., Taylor, D. and Bright, J. (2020), "What can forensic probabilistic genotyping software developers learn from significant non-forensic software failures?", *WIREs Forensic Science,* Vol. 3, No. 2, 8 pp.

Budiman, E., Wati, M., Widians, J. A., Puspitasari, N., Firdaus, M. B. and Alameka, F. (2018), "ISO/IEC 9126 Quality Model for Evaluation of Student Academic Portal", *Proceeding of EECSI, Malang - Indonesia,* pp. 78 – 83.

Bychkov, D. (2013), "Desktop verses Web Applications: A Deeper Look and Comparison", https://www.seguetech.com/desktop-vs-web-applications/, Accessed: May 25, 2021.

Campbell, D. (2019), “The many human errors that brought down the Boeing 737 Max”, https://www.theverge.com/2019/5/2/18518176/boeing-737-max-crash-problems-human-error-mcas-faa, Accessed: March 25, 2021.

Cankaya, E. C. (2011), “Bell-LaPadula Confidentiality Model”, In *Encyclopedia of Cryptography and Security,* Van-Tilborg, H. C. A. and Jajodia, S. (ed.), 2nd edition, Vol. 1, Springer, Boston, MA, 1496 pp.

Chetan, K. (2017), *Securing Node Applications*, O'Reilly Media, Inc, 210 pp.

Choudhuri, P. K. (2014), “Application of Multi-Criteria Decision Making (MCDM) Technique for Gradation of Jute Fibres”, *Journal of The Institution of Engineers (India): Series E,* Vol. 95, pp. 63–68.

Choudhury, M. M. and Choudhury, A. M. (2010), “Identification of the characteristics of E-commerce websites”, *Webology Journal*, Vol. 7, No. 1, 10 pp.

Christakis, M. and Bird, C. (2016), “What Developers Want and Need from Program Analysis: An Empirical Study”, *31st IEEE/ACM International Conference on Automated Software Engineering (ASE),* pp. 332 - 343.

Crosby, P. B. (1979), “Quality is free- if you understand it”, *Inc. Journal,* 4 pp.

Deming, W. E. (1986), *Out of the Crisis*, Massachusetts Institute of Technology, Cambridge.

Derisma, D. (2020), “The Usability Analysis Online Learning Site for Supporting Computer programming Course Using System Usability Scale (SUS) in a university”, *International Journal of Interactive Mobile Technologies*, Vol. 14, No. 9, pp. 182 – 195.

Djouab, R. and Bari, M. (2016), “An ISO 9126 Based Quality Model for the e-Learning Systems”, *International Journal of Information and Education Technology*, Vol. 6, No. 5, 6 pp.

Dorri, A., Kanhere, S. S., Jurdak, R. and Gauravam, P. (2017), “Blockchain for IoT security and privacy: The case study of a smart home”, *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops),* 25 pp.

Dromey, G. R. (1995), “A Model for Software Product Quality”, *IEEE Transaction on Software Engineering,* Vol. 21, No. 2*,* 9 pp.

Dubey, S. K. and Mishra, A. (2014), “Fuzzy Qualitative Evaluation of Reliability of Object-Oriented Software System”, *IEEE International Conference on Advances in Engineering and Technology Research (ICAETR),* pp. 1–6.

Dubey, S. K., Ghosh, S. and Rana, A. (2012), “Comparison of Software Quality Models: An Analytical Approach”, *International Journal of Emerging Technology and Advanced Engineering,* Vol. 2, No. 2, 9 pp.

Duke, S. O. O. and Obidinnu, J. N. (2010), “An improved COCOMO software cost estimation model”, *Global Journal of Pure and Applied Sciences*, Vol. 16, No. 4, pp. 479 – 492.

Dwivedi, S. and Dubey, S. K. (2014), “Measurement of Web Usability: An Approach”, *International Journal of Computer and Communication System Engineering,* pp. 59-65.

Fahmy, S., Haslinda, N., Roslina, W. and Fariha, Z. (2012), “Evaluating the Quality of Software in E-Book using the ISO 9126 Model”, *International Journal of Control and Automation*, Vol. 5, No. 2, 8 pp.

Fawareh, H. (2020), “Software Quality Model for Maintenance Software Purposes”, *International Journal of Engineering Research and Technology*, Vol. 13*,* No. 1, pp. 158 – 162.

Febrero, F., Moraga, M. A. and Calero, C. (2017), “Software Reliability as User Perception Application of the Fuzzy Analytic Hierarchy Process to Software Reliability Analysis”, *IEEE International Conference on Software Quality, Reliability and Security (QRS),* pp. 224–231.

Feigenbaum, A. V. (1991), *Total Quality Control*, 3rd ed., McGraw-Hill, New York.

Felderer, M., Büchler, M., Johns, M., Brucker, A. D., Breu, R. and Pretschner, A. (2016), "Security Testing: A Survey", *Journal of Advances in Computers*, Vol. 101, pp. 1 - 51.

Fleischman, W. and Crawford, J. (2020), "Once Again, We Need to Ask, what have We Learned from Hard Experience?", In *Societal Challenges in the Smart Society*, International Conference on the Ethical and Social Impact of ICT, Mario, A., Jorge, P. Kiyoshi, M. and Palma, A. M. L. (ed.), pp. 561 – 571.

Galli, T., Chiclana, F. and Siewe, F. (2020), "Software Product Quality Models, Developments, Trends, and Evaluation", *SN Computer Science*, Vol. 1, No. 154, 24 pp.

Gambo, I., Soriyan, A. and Achimugu, P. (2011), "Software Architecture Performance Quality Model: Qualitative Approach", *ARPN Journal of Systems and Software*, Vol. 1, No. 1, pp. 28-33.

Georgiadou, E. (2003), "GEQUAMO– A Generic, Multilayered, Customisable, Software Quality Model", *International Journal of Cybernetics*, Vol. 11, No. 4, pp. 313-323.

Ghezzi, C., Jazayeri, M. and Mandrioli, D. (1991), *Fundamentals of Software Engineering*, Pearson Publishing, 604 pp.

Ghosh, N., Singhal, R. and Das, S. K. (2019), "A Risk Quantification Framework to Authorise Requests in Web-based Collaborations", *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pp. 247-254.

Gorrie, M. (2021), "Sensitive data exposure: What is it and how it's different from a data breach", https://us.norton.com/internetsecurity-privacy-sensitive-data-exposure-how-its-different-from-data-breach.html, Accessed: May 28, 2021.

Grady, R. B. (1992), *Practical Software Metrics for Project Management and Process Improvement*, Prentice Hall, Englewood Cliffs, NJ, USA, 282 pp.

Gupta, S. and Gupta, B. B. (2017), "Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art", *International Journal of System Assurance Engineering and Management*, Vol. 8, pp. 512 – 530.

Hamit, J. (2014), "Top Ten Web Security Risks: Missing Function Level Access Control (#7)", https://www.credera.com/insights/top-ten-web-security-risks-missing-function-level-access-control-7, Accessed: May, 28, 2021.

Hana, R. I., Abeer, S. J. and Hana, E. (2019), "Software Engineering Cost Estimation Using COCOMO II Model", *African Journal Online,* 26 pp.

Hassan, M. M., Nipa, S. S. Akter, M., Haque, R., Deepa, F. N., Rahman, M., Siddiqui, A. and Sharif, H. M. (2018), "Broken Authentication and Session Management Vulnerability: A Case Study of Web Application", *International Journal of Simulation - Systems, Science and Technology,* Vol. 19, No. 2, 11 pp.

Henk, C. A. T. and Sushil, T. (2014), *Encyclopaedia of Cryptography and Security,* Springer Science and Business Media, 1457 pp.

Hopkins, S., Henry, C., Bagui, S., Mishra, A., Kalaimannan, E. and John, C. S. (2020), "Applying a Verified Trusted Computing Base to Cyber Protect a Vulnerable Traffic Control Cyber-Physical System", *IEEE Southeast Conference,* 8pp.

Hussain, S., Farid, S. and Mumtaz, I. (2019), "Is Customer Satisfaction Enough for Software Quality?", *International Journal of Computer Science and Software Engineering (IJCSSE),* Vol. 8, No. 2, pp. 40-47.

Ibanga, I. (2021), "Steps to Build a School Management Portal in Nigeria, Things Needed, Cost, Benefits", https://infoguidenigeria.com/steps-to-build-a-school-management-portal-in-nigeria/, Accessed: March 20, 2021.

Ishikawa, K. (1989), "How to apply companywide quality control in foreign countries", *Quality Progress*, Vol. 22, No. 9, pp. 70 - 74.

Islam, A. and Tsuji, K. (2011), "Evaluation of Usage of University Websites in Bangladesh", *DESIDOC Journal of Library and Information Technology*, Vol. 31, No. 6, pp. 469 - 479.

Jamwal S. R. and Jamwal D. (2009), "Issues and Factors for Evaluation of Software Quality Models", *Proceedings of the 3rd National Conference, INDIACom*, 9 pp.

Johns, M., Engelmann, B. and Posegga, J. (2008), "XSSDS: Server-side Detection of Cross-site Scripting Attacks", *Annual Computer Security Applications Conference,* pp. 335 – 344.

Johnson, M. (2016), *Cyber Crime Security and Digital Intelligence,* Routledge Publishers, 2 Park Square, Milton Park, Abingdon, Oxon OX14 4RN, 304 pp.

Johnston, P. (2021), "Historical Software Accidents and Errors", https://embeddedartistry.com/fieldatlas/historical-software-accidents-and-errors/, Accessed: 17 April, 2021.

Juran, J. M. (1988), *Juran on planning for quality,* New York free Press, London, 341 pp.

Justiniano, I. (2015), "Security Models: Integrity, Confidentiality and Protection of the Data", https://www.linkedin.com/pulse/security-models-integrity-confidentiality-protection-data-justiniano/, Accessed: May 12, 2021.

Kabir, M. A., Rehman, M. and Majumdar, S. I. (2016), "An Analytical and Comparative Study of Software Usability Quality Factors", *7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 800 - 803.

Kasisopha, N., Rongviriyapanish S. and Meananeatra, P. (2020), "Method Evaluation for Software Testability on Object Oriented Code", *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pp. 308 - 313.

Kassie, N. B. and Singh, J. (2020), "A study on software quality factors and metrics to enhance software quality assurance", *International Journal of Productivity and Quality Management,* Vol. 29, No. 1, pp. 24–44.

Kaur, S. (2012), "Software Quality", *International Journal of Computers and Technology*, Vol. 3, No. 1, pp. 127 – 131.

Kaur, S., Kaur, K. and Kaur, P. (2016), "An Empirical Performance Evaluation of Universities Website", *International Journal of Computer Applications,* Vol. 146, No. 15, pp. 10 – 16.

Kelley, P. G., Komanduri, S., Mazurek, M. L., Shay, R., Vidas, T., Bauer, L. and Lopez, J. (2012), “Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms”, *IEEE Symposium on Security and Privacy*, pp. 523 - 537.

Khwanruthai, B. (2012), “How to do AHP Analysis in Excel”, *Division of Spatial Information Science Presentation Slides for Graduate School of Life and Environmental Sciences*, University of Tsukuba, 21pp.

Kitchenham, B. and Pickard, L. (1989), “Towards a Constructive Quality Model. Part 2: Statistical Techniques for Modelling Software Quality in the ESPRIT REQUEST Project”, *Software Engineering Journal*, pp 114 -126.

Kokalitcheva, K. (2015), “Drivers’ Licenses and Social Security Numbers were leaked online by Uber”, www.fortune.com, Accessed: June 26, 2018.

Kous, K., Pusnik, M., Hericko, M. and Polancic, G. (2018), “Usability Evaluation of a Library Website with Different End User Groups”, *Journal of Librarianship and Information Science*, pp. 1 – 16

Kudkar, I. (2021), “OWASP: Sensitive Data Exposure Attacks”, https://medium.com/shallvhack/owasp-sensitive-data-exposure-attacks-7ef41e6b4a59, Accessed: May 28, 2021.

Kumar, A., Sah, B., Singh, A. R., Deng, Y., He, X. and Kumar, P. (2017) “A review of multi criteria decision making (MCDM) towards sustainable renewable energy development”, *Journal of Renewable and Sustainable Energy Reviews*, Vol. 69, pp. 596 - 609.

Kumar, P. and Singh, S. K. (2016), “A Comprehensive Evaluation of Aspect-Oriented Software Quality (AOSQ) Model using Analytic Hierarchy Process (AHP) Technique”, *2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Fall)*, 7 pp.

Kurt, S. (2011), “The accessibility of university web sites: the case of Turkish universities”, *Universal Access in Information Society*, Vol. 10, No. 1, pp. 101 – 110.

Lai, V. S., Wong, B. K. and Cheung, W. (2002), “Group decision making in a multiple criteria environment: A case using the AHP in software selection”, *European Journal of Operational Research*, Vol. 137, pp. 134–144.

Lai, Y. and Ishizaka, A. (2019), “The application of multi-criteria decision analysis methods into talent identification process: A social psychological perspective”, *Journal of Business Research. Elsevier*, Vol. 109, pp. 637-647.

Lisa, C. (2001), “Is quality negotiable?”, *STARWest Software Testing Conference*, 2 pp.

Liu, G., Wang, C., Zhang, R., Wang, Q., Song, H. and Ji, S. (2017), “BTG-BIBA: A Flexibility-Enhanced Biba Model Using BTG Strategies for Operating System”, *International Journal of Computer and Information Engineering*, Vol. 11, No. 6, pp. 765–771.

Liu, H., Dai, Z., Li, J. and Zhou, Y. (2016), “An Improved MLS Policy Model”, *10th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, pp. 47–52.

Madan, A. and Dubey, S. K. (2012), “Usability Evaluation Methods: a literature review”, *International Journal of Engineering Science and Technology (IJEST)*, Vol. 4, No. 2, pp. 590-599.

Mahmudova, S. and Jabrailova, Z. (2020), “Development of an algorithm using the AHP method for selecting software according to its functionality”, *Journal of Soft Computing*, Vol. 24, pp. 8495 - 8502.

Martin, S. (2020), “Create A Social Media App & How Much It Costs In 2021”, https://medium.com/flutter-community/build-a-social-media-mobile-app-its-cost-features-business-model-etc-62718c7d05e4, Accessed: March 20, 2021.

Martins, J., Bezerra, C., Uchoa, A. and Garcia, A. (2020), “Are Code Smell Co-occurrences Harmful to Internal Quality Attributes? A Mixed-Method Study”, *Proceedings of the 34th Brazillian Symposium on Software Engineering*, pp. 52 – 61.

Maryoly, O., Perez, M. A. and Rojas, T. (2002), “A Systemic Quality Model for Evaluating Software Products”, *Investigative Laboratory and Information System*, 6 pp.

McCall, J. A., Richards, P. K., and Walters, G. F. (1977), “Factors in Software Quality”, *RADC TR, US Rome Air Development Centre Reports*, Vol. 1, pp. 77-369,

McMillin, B and Roth, T. (2017), *Cyber-Physical Security and Privacy in the Electric Smart Grid*, Morgan and Claypool Publishers, 66 pp.

Mishra, A. and Otaiwi, Z. (2020), “DevOps and software quality: A systematic mapping”, Computer Science Review, Vol. 38, 13 pp.

Moe, E. E. and Thwin, M. M. S. (2019), “Effective Security and Access Control Framework for Multilevel Organisations”, *Advances in Biometrics*, pp 267-288.

Mohammed, A. A., Ruzaini, A. A., Emad, A. and Nabil, E. (2016), “The Effect of Security and Privacy Perceptions on Customers Trust to Accept Internet Banking Services: An Extension of TAM”, *Journal of Engineering and Applied Sciences*, Vol. 11, No. 3, pp. 545 - 552.

Mohino, J. V., Higuera, J. B., Higuera, J. R. B. and Montalvo, J. A. S. (2019), “The Application of a New Secure Software Development Life Cycle (S-SDLC) with Agile Methodologies”, *State of the Art of Cyber Security*, Vol. 8, No. 11, pp. 12-18.

Morgenstern, M., Marx, A. and Landesman, M. (2005), “Insecurity in Security Software”, *Virus Bulletin Conference*, pp. 212-221.

Nasrabadi, M. Z. and Parsa, S. (2021), “Learning to Predict Software Testability”, *26th International Computer Conference, Computer Society of Iran (CSICC)*, pp. 1 – 5.

Nielsen, J. (2003), “Usability 101: Introduction to Usability,” http://www.ingenieriasimple.com/usabilidad/IntroToUsability.pdf, Accessed: March 22, 2021.

Nielsen, J. (2012), “Usability 101: Introduction to Usability,” https://www.nngroup.com/articles/usability-101-introduction-to-usability/, Accessed: March 22, 2021.

Nihal, K. and Abran, A. (2001), “Analysing Measuring and Assessing Software Quality Within a Logic-Based Graphical Framework”, *International Conference on Structural Dynamics,* 8 pp.

Nilson, M., Antinyan, V. and Gren, L. (2019), “Do Internal Software Quality Tools Measure Validated Metrics?”, *International Conference on Product-Focused Software Process Improvement,* pp. 637 – 648.

Nistala, P., Nori, K. V. and Reddy, R. (2019), “Software Quality Models: A Systematic Mapping Study”, *IEEE/ACM International Conference on Software and System Processes*, pp.125-134.

Noe, E. (2017), “Usability, Accessibility and Web Security Assessment of E-government Websites in Tanzania”, *International Journal of Computer Applications*, Vol. 164, No 5, pp. 42 – 48.

O’Shea, M. (2017), “How does a web application work”, https://www.quora.com/What-is-a-web-application-3, Accessed: April 8, 2021.

Ochaun, M. (2020), “Using Components with Known Vulnerabilities”, https://securityboulevard.com/2020/04/using-components-with-known-vulnerabilities-2/, Accessed: May 29, 2021.

Ogundele, L. A. (2018), “Software Quality Assurance using Adaptive Agent-based Cleanroom Approach”, *Unpublished PhD Thesis Report,* Federal University of Technology, Akure, 190 pp.

Okudan, O. and Budayan, C. (2020), “Assessment of Project Characteristics Affecting Risk Occurrences in Construction Projects using Fuzzy AHP”, Sigma Journal of Engineering and Natural Science, Vol. 38, No. 3, pp. 1447 – 1462.

Olav, L. (2018), *The Huawei and Snowden Questions,* Springer International Publishing, 123pp.

Omar, S. and Fahad, A. (2017), "Integrating Knowledge Life Cycle within Software Development Process to Produce a Quality Software Product", *International Conference on Engineering and Technology (ICET),* pp. 1 – 7.

Osman T., Adballah, O. B., Reda, M. S. A., Mohammed, R. and Kabli, A. (2014), "Construction Projects Selection and Risk Assessment by Fuzzy AHP and Fuzzy TOPSIS Methodologies", *Applied Soft Computing*, Vol. 17, pp. 105-116.

Ouissem, B. F., Omar, C., Moez, K., Habib, H. and Abdelouahib, D. (2021), "An OWASP Top Ten Driven Survey on Web Application Protection Method", *International Conference on Risks and Security of Internet and Systems,* Vol. 34, No. 1, 16 pp.

Padayachee, I., Kotze, P. and Van-Der-Merwe, A. (2010), "ISO 9126 external systems quality characteristics, sub-characteristics and domain specific criteria for evaluating E-Learning systems", *Conference of the Southern African Computer Lecturers Association, South Africa*, 9 pp.

Panagiotou, D. and Mentzas, G. (2011), "Leveraging Software Reuse with Knowledge Management in Software Development", International Journal of Software Engineering and Knowledge Engineering, Vol. 21, No. 5, pp. 693 – 723.

Pandya, D. and Patel, N. J. (2016), "OWASP Top 10 Vulnerability Analyses in Government Websites", *International Journal of Enterprise Computing and Business Systems*, Vol. 6, No. 1, pp. 1 - 18.

Parthasarathy, S., Sridharan, C., Chandrakumar, T. and Sridevi, S. (2020), "Quality Assessment of Standard and Customised COTS Products", *International Journal of Information Technology Project Management*, Vol. 11, No. 3, pp. 1–13.

Patel, S. and Sahani, G. J. (2018), "A Survey on Different Type of Access Control Model for Personal Health Record (PHR) System", *International Journal of Research in Engineering, Science and Management*, Vol. 1, No. 9, pp. 380 - 384.

Pedamkar, P. (2020), "What is Application Software and Its Types", https://www.educba.com/what-is-application-software-its-types/, Accessed: March 20, 2021.

Petersen, J. (2021), "The United States air accident investigator says metal fatigue is the likely cause of an engine fire on a Boeing 777 aircraft last week", https://search.informit.org/doi/full/10.3316/TVNEWS.TSM202102230238, Accessed: March 7, 2021.

Poggi, A. (2018), "Information Attacks and Defenses on the Social Web", *Global Implications of Emerging Technology Trends,* 20 pp.

Pohl, C., and Hof, H. J. (2015), "Secure scrum: development of secure software with scrum", *The Ninth International Conference on Emerging Security Information, Systems and Technologies – SECURWARE,* 7 pp.

Polat, G., Damci, A., Turkoglu, H. and Gurgun, A. P. (2017), "Identification of root causes of construction and demolition (C&D) waste: The case of Turkey", *Creative Construction Conference,* pp. 948 – 955.

Qui, Y. F. Chui, Y. P. and Helander, M. G. (2006), "Usability Analysis of Mobile Phone Camera Software Systems", *IEEE Conference on Cybernetics and Intelligent Systems,* pp. 1–6.

Quirchmayr, G., Funilkul S. and Chutimaskul, W. (2007), "A Quality Model of E Government Services Based on the ISO/IEC 9126 Standard", *Proceedings of International Legal Informatics Symposium, IRIS,* 8 pp.

Rahardjo, E., Mirchandani, D. and Joshi, K. (2014), "E-Government Functionality and Website Features: A Case Study of Indonesia", *Journal of Global Information Technology Management*, pp. 31-50.

Ramkumar, M. (2017), Minimal TCB for System-Model Execution, *The 2017 International Conference on Security and Management*, 7 pp.

Regan, G., McCaffery, F., Chandra, P. P., Reich, J., Armengaud, E., Kaypmaz, C., Guo, Z. J., Zeller, M., Longo, S. and O'Carroll, E. (2020), "Quality improvement mechanism for cyber physical systems - An evaluation", *Journal of Software: Evolution and Process*, Vol. 32, No. 11, 11 pp.

Rehman, A. (2019), "How Much Does It Cost to Build a Mobile-App like NatWest Online Banking App?", https://www.branex.co.uk/blog/how-much-does-it-cost-to-build-a-mobile-app-like-natwest-online-banking/, Accessed: March 20, 2021.

Saaty, T. L. (1977), "A scaling method for priorities in hierarchical structures", *Journal of Mathematical Psychology*, Vol. 15, No. 3, pp. 234 – 281.

Saaty, T. L. (2008), "Decision making with the analytic hierarchy process", *International Journal of Services Sciences*, Vol. 1, No. 1, pp. 83 – 98.

Sahu, K. and Srivastava, R. K. (2018), "Soft Computing Approach for Prediction of Software Reliability", *ICIC Express Letters ICIC International*, Vol. 12, No. 12, pp. 1213–1222.

Saini, G. L., Panwar, D., Kumar, S. and Singh, V. (2020), "A systematic literature review and comparative study of different software quality models", *Journal of Discrete Mathematical Sciences and Cryptography*, Vol. 23, No. 2, pp. 585 – 593.

Salleh, M. A., Bahari, M. and Zakaria, N. H. (2017), "An Overview of Software Functionality Service: A Systematic Literature Review", *Procedia Computer Science*, Vol. 124, pp. 337–344.

Salman, O., Kayssi, A., Chehab, A. and Elhajj, I. (2017), "Multi-Level Security for the 5G/IoT Ubiquitous Network", *Second International Conference on Fog and Mobile Edge Computing (FMEC), IEEE*, pp. 188–193.

Salve, M. S., Samreen, S. N. and Khatri-Valmik, N. (2018), "A Comparative Study on Software Development Life Cycle Models", *International Research Journal of Engineering and Technology (IRJET)*, Vol. 5, No. 2, pp. 696 – 700.

Saravanan, N. and Umamakeswari, A. (2020), "Lattice Based Access Control for Protecting User Data in Cloud Environments with Hybrid Security", *Journal of Computers and Security*, 23 pp.

Sarkar, B. (2011), "Fuzzy decision making and its applications in cotton fibre grading", In *Woodhead Publishing Series in Textiles: Soft Computing in Textile Engineering*, Woodhead Publishing, Majumdar, A. (eds.), pp. 353 – 383.

Schinagl, S., Paans, R. and Schoon, K. (2016), "The Revival of Ancient Information Security Models, Insight in Risks and Selection of Measures", *49th Hawaii International Conference on System Sciences*, Vol. 1, pp. 4041 – 4050.

Sharma, A., Kumar, R. and Grover, P. S. (2020), "Managing Component-Based Systems with Reusable Components*", International Journal of Computer Science and Security*, Vol. 1, No. 2, pp. 52 – 57.

Sharma, C. and Dubey, S. K. (2015), "A Perspective Approach of Software Reliability Models and Techniques", *ARPN Journal of Engineering and Applied Sciences*, Vol. 10, no. 16, pp. 7300 - 7308.

Sharma, M. K. (2017), "A study of SDLC to develop well engineered software", *International Journal of Advanced Research in Computer Science*, Vol. 8, No. 3, pp. 520 – 523.

Shasha, S., Mahmoud, M., Mannan, M. and Youssef, A. (2019), "Playing with Danger: A Taxonomy and Evaluation of Threats to Smart Toys", *IEEE Internet of Things Journal*, Vol. 6, No. 2, pp. 2986 - 3002.

Shewhart, W. A. (1931), *Economic Control of Quality Manufactured Product*, Van Nostrand Publishing, New York.

Siavvas, M. G., Chatzidimitriou, K. C. and Symeonidis, A. L. (2017), "QATCH - An adaptive framework for software product quality assessment", *Expert Systems with Applications*, Vol. 86, pp. 350 – 366.

Signore, O. (2005), "A comprehensive model for Web sites quality", *Seventh IEEE International Symposium on Web Site Evolution,* pp. 30-36

Singh, P., Thevar, K., Shetty, P. and Shaikh, B. (2015), "Detection of SQL Injection and XSS Vulnerability in Web Application", *International Journal of Engineering and Applied Sciences (IJEAS),* Vol. 2, Issue 3, pp. 16 – 21.

Smith, J. (2021a), "Desktop Applications Vs. Web Applications", https://www.streetdirectory.com/travel_guide/114448/programming/desktop_applications_vs_web_applications.html, Accessed: May 20, 2021.

Smith, J. (2021b), "eCommerce Website Pricing: Determining Cost of an eCommerce Build", https://www.outerboxdesign.com/web-designarticles/ecommerce_website_ pricing, Accessed: March 20, 2021.

Srinivasan, S. M. and Sangwan, R. S. (2017), "Web App Security: A Comparison and Categorisation of Testing Frameworks", *IEEE Software Journal*, Vol. 34, No. 1, pp. 99-102.

Sudhodanan, A., Carbone, R., Compagna, L., Dolgin, N., Armando, A. and Morelli, U. (2017), "Large-Scale Analysis & Detection of Authentication Cross-Site Request Forgeries", *IEEE European Symposium on Security and Privacy (EuroS&P),* pp. 350-365.

Sukmasetya, P., Setiawan, A. and Arumi, E. R. (2020), "Usability evaluation of university website: a case study", *Journal of Physics: Conference Series*, Vol. 1517, 6 pp.

Sundar, V. (2014), "OWASP A9 Using Components with Known Vulnerabilities", https://www.indusface.com/blog/components-known-vulnerabilities/, Accessed: May 29, 2021.

Suveetha, K. and Manju, T. (2016), "Ensuring confidentiality of cloud data using homomorphic encryption", *Indian Journal of Science and Technology,* Vol. 9, No. 8, 7 pp.

Tabassum, A., Bhatti, N. S., Asghar, R. A., Manzoor, I. and Alam, I. (2017), "Optimised Quality Model for Agile Development: Extreme Programming (XP) as a Case Scenario", *(IJACSA) International Journal of Advanced Computer Science and Applications*, Vol. 8, No. 4, 9 pp.

Teknomo, K. (2017), "Analytic hierarchy process (AHP) tutorial", https://people.revoledu.com/kardi/tutorial/AHP/, Accessed: March 25, 2021.

Thamer, A. A., Mohammad, M. and Ahmad, A. (2013), "Evaluating the Quality of Software in ERP Systems Using the ISO 9126 Model", *International Journal of Ambient Systems and Applications (IJASA)*, Vol. 1, No. 1, pp. 1-9.

Thomas, M. (2020), "Software 101: A Complete Guide to the Different Types of Software", https://www.coderus.com/software-101-a-complete-guide-to-the-different-types-of-software/, Accessed: March 20, 2021.

Tinnaluri, V. S, N. (2016), "A Panorama of Quality Assurance (QA) In Software Appliances", *International Journal of Current Research and Academic Review*, Vol. 4, No. 6, 9 pp.

Toapanta, M., Nazareno, J., Tingo, R., Mendoza, F., Orizaga, A. and Mafla, E. (2018), "Analysis of the Appropriate Security Models to Apply in a Distributed Architecture", *IOP Conference Series: Materials Science and Engineering*, Vol. 423, 6 pp.

Tomov, L. and Ivanova, V. (2015), "Software Quality from Systems Perspective", *Proceedings of the 11th Annual International Conference on Computer Science and Education in Computer Science*, 10 pp.

Tripathi, S. (2014), "A Survey on Quality Perspective and Software Quality Models", *IOSR Journal of Computer Engineering (IOSR-JCE)*, Vol. 16, No. 2, pp. 63-72.

Uska, M. Z., Wirasasmita, R. H. and Fahrurrozi, M. (2019), "The application of Usability Testing Method for Evaluating the New Student Acceptance (NSA) System" *Journal of Physics: Conference Series*, Vol. 1539, 6 pp.

Valenti, S., Cucchiarelli, A. and Panti, M. (2002), “Computer Based Assessment Systems Evaluation via the ISO9126 Quality Model”, *Journal of Information Technology Education,* Vol. 1, No. 3, pp. 157-175.

Verma, S. and Mehlawat, M. K. (2017), “Multi-criteria Optimisation model integrated with AHP for evaluation and selection of COTS components”, *Special Issue on Advances in Optimisation Theory and Applications on the occasion of the International Conference on Recent Advances in Optimisation Theory and Applications – RAOTA,* Vol. 66, No. 11, pp. 1879 - 1894, pp. 1–16.

Waliaro, D. O., Omieno, K. and Ondulo, J. (2019), “Analysis of Software Quality Models for ERP Software Use in University in Kenya”, *International Journal of Advanced Research in Computer and Communication Engineering,* Vol. 8, No. 10, pp. 58–61.

Wang, C., Tsai, H., Ho, T., Ngugen, V. and Huang, Y. (2020), “Multi-Criteria Decision Making (MCDM) Model for Supplier Evaluation and Selection for Oil Production Projects in Vietnam”, *Journal of Multi-Objective Optimisation of Processes,* Vol. 8, No. 2, 13pp.

Weichbroth, P. (2018), “Usability Attributes Revisited: A Time-Framed Knowledge Map”, *Federated Conference on Computer Science and Information Systems (FedCSIS)*, 25 pp.

Weiss, A., Gautham, S., Jayakumar, A. V., Elks, C. R., Kuhn, R., Kacker, R. N. and Preusser, T. B. (2021), “Understanding and Fixing Complex Faults in Embedded Cyberphysical Systems”, *Journal of Computers,* Vol. 54, No. 1, pp. 49 – 60.

Westmacott, M. (2019), “Biba Security Model Inspired Social Media Security Controls”, *Conference for Truth and Trust Online,* 2 pp.

Yadav, A. and Shah, R. (2015), “Review on Database Access Control Mechanisms and Models”, *International Journal of Computer Applications*, Vol. 120, No. 18, 4pp.

Yadav, S. and Kishan, B. (2020), “Analysis and Assessment of Existing Software Quality Models to Predict the Reliability of Component-Based Software”, *International Journal of Emerging Trends in Engineering,* Vol. 8, No. 6, 17 pp.

Yujun, M., Lingli, L., Shuaijun, D. and Guofeng, L. (2019), "AHP-Based Software Quality Risk Assessment Method for Information System", *Scientific Conference on Network, Power Systems and Computing (NPSC)*, pp. 189–193.

Zeng, G. (2019), "On the confusion matrix in credit scoring and its analytical properties", *Journal of Communications in Statistics - Theory and Methods*, Vol. 49, No. 9, pp. 2080 – 2093.

Zhang, J., Hu, H. and Huo, S. (2021), "A Browser-based Cross Site Request Forgery Detection Model", *2nd International Conference on Electronics and Communication, Network and Computer Technology (ECNCT)*, Chengdu, China, 5 pp.

Zhu, D., Yang, Y., Jin, H., Shao, J. and Feng, W. (2016), "Application of Modified BLP model on Mobile Web Operating System", *IEEE TrustCom/BigDataSE/ISPA*, pp. 1818-1825.

APPENDICES

APPENDIX A CODES USED

```
<!doctype html>

<html>

<head>

   <base href="/">

   <meta charset="utf-8" />

   <link rel="icon" type="image/png" href="assets/img/favicon.ico">

   <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

   <title>Software Assurance Test</title>

   <!-- Animate -->

   <link      href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.2.0/animate.min.css"
rel="stylesheet">

   <style media="screen">

   #loader {  position: absolute;

     display: block;

     left: 0;

     right: 0;

     margin-left: auto;

     margin-right: auto;

     top: 50%; }

   </style>

</head>

<body>

  <app-root>

     <div id="loader"><img src="assets/img/loader-preview.svg" alt="loading"></div>
  </app-root>
</body>
</html>
import {Request, Response} from "express";
const axios = require("axios");
const webdriver = require("selenium-webdriver"),
  By = webdriver.By,
  until = webdriver.until;
const Key = require("selenium-webdriver");
const gtmetrix = require("gtmetrix")({
  email: "admin@rudefish.wtf",
  apikey: "0af802c318438f2783d59121d373a7a7",});
export const testComponent = {
  runTest: (req: Request, res: Response) => {
    const sites = [
      req.body.site1,
      req.body.site2,
      req.body.site3,
      req.body.site4, ];
    let results: any = [];
    let sendRequest = (site: any) => {
      return new Promise((resolve: any, reject: any) => {
        const testDetails = {
          url: `${site}`,
          location: 1,
          browser: 3,};
        gtmetrix.test.create(testDetails).then((data: any) => {
          gtmetrix.test.get(
            data.test_id,
            5000,
            (error: Error, response: any) => {
              if (response) {
                results.push(response);
                resolve(response); } else {
                console.log(error.message);
                reject(error);}}
        });
let sendAllRequests = async () => {
      let i = 0;
      for (let site of sites) {
        try {   await sendRequest(site);
          i += 1;
          if (i == 4) {
            return res.status(200).json(results); }
        } catch (e: any) {
          i += 1;
          return res.status(500).send(e);}} };
    sendAllRequests();},
  makeGetRequests: (req: Request, res: Response) => {
    const sites = [
      req.body.site1,
      req.body.site2,
      req.body.site3,
      req.body.site4, ];
    let results: Array<any> = [];
    let sendRequest = (site: any, times: number) => {
      return new Promise((resolve: any, reject: any) => {
        let totalRequests = times;
        let sReq = 0;
        let sErr = 0;
        let miniPromise = () => {
          return new Promise((resolve: any, reject: any) => {
            for (let i = 0; i < totalRequests; i++) {
              axios
                .get(`${site}`)
                .then((response: any) => {
                  if (response) {
                    sReq += 1;
                  } else {
                    sErr += 1;}
                  if (i == totalRequests - 1) {
                    resolve(response);}
                })
                .catch((error: Error) => {
                  sErr += 1;
                  if (i == totalRequests - 1) {
                    reject(error);}
                });
        let fireMiniPromise = async () => {
          try {
            await miniPromise();
            const dataToPush = {
              totalRequests: totalRequests,
              success: sReq,
              failure: totalRequests - sReq, };
            results.push(dataToPush);
            resolve(dataToPush);
          } catch (error: any) {
            reject(error);} };
        fireMiniPromise();
      }); };
    let sendGetRequests = async () => {
      let i = 0;
      for (let site of sites) {
        try {
          await sendRequest(site, Number(req.body.no_of_times));
          i += 1;
          if (i == 4) {
            return res.status(200).json(results); }
        } catch (error: any) {
          i++;
          return res.status(500).send(error); } };
    sendGetRequests(); },
```