Automated User Interface Usability Evaluation Framework for Android Mobile Applications

Name of Candidate: Zeleke Chekol

A Thesis Submitted to the

Department of Computing

School of Electrical Engineering and Computing

Presented in Partial Fulfillment for the Degree of Master of Science in Software Engineering

Office of Graduate Studies

Adama Science and Technology University

Adama

January, 2018

Automated User Interface Usability Evaluation Framework for Android Mobile Applications

Name of Candidate: Zeleke Chekol

Name of Advisor: Dr. Mesfin Abebe

A Thesis Submitted to the

Department of Computing

School of Electrical Engineering and Computing

Presented in Partial Fulfillment for the Degree of Master of Science in Software Engineering

Office of Graduate Studies

Adama Science and Technology University

Adama

January, 2018

Declaration

I hereby declare that this MSc Thesis is my original work and has not been presented for a degree in any other university, and all sources of material used for this thesis have been duly acknowledged.

Name: _____

Signature: _____

This MSc Thesis has been submitted for examination with my approval as thesis advisor.

Name: _____

Signature: _____

Date of submission: _____

Approval of Board of Examiners

We, the undersigned, members of the Board of Examiners of the final open defense by _____ have read and evaluated his/her thesis entitled "_____" and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirement of the Degree of ……

_____	_____	_____
Supervisor /Advisor	Signature	Date
_____	_____	_____
Chairperson	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Acknowledgement

I am gratefully indebted to the almighty God and St. Marry whose everlasting and undying love kept me and for infinite mercy throughout my life and during the course of this thesis work. Next, I would like to express my deep and heart full gratitude to my advisor Dr. Mesifn Abebe for his kindly approach, inevitable support, understanding and critical concerns, inevitable comment started from the proposal up to accomplish of this thesis work.

My special thanks also go to the staff members of computing department; they provided me the knowledge and supported me not only during my Masters Study but also BSc study.

I would like to acknowledge ASTU particularly department of computing for the provision of the computer lab with internet connection. I have to acknowledge and appreciate my all beloved friends specially Daikon Adane and Kelemework both are supported in one way or the other to the fulfillment of this research work. I have to acknowledge all lab mates in Smart Software and System Lab. I am also grateful for Ato Yehunie, CSE store administrator for his facilitating the utilization of the lab.

Last but not least, my immeasurable thanks go to my wonderful and caring parents specially my mother Tiringo Abawa , for their support everything from I was a child up to now, my beloved uncle Sagin Mengestie Abawa , for his financial, idea, and every support through my life. Without your advice and follow-up I am not reach this day because you are inspire and encourage me all the times especially at the most difficult and seemingly hopeless situations. Thank you my all parents for your love, moral, and financial support towards my education that I cannot list you all because of limited space.

Table of Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations and Acronyms

Abbreviation	*Expanded form*
AAS	Actual Action Sequence
AIM	Automated Inspection Method
App	Application
ASTU	Adama Science and Technology University
AUEM	Automated Usability Evaluation method
AUT	Automated Usability Testing
AUTMA	Automated Usability Testing for Mobile Applications
BES	Blackberry Enterprise Server
BIS	Blackberry Internet Service
CSS	Cascading Sheet Style
CW	Cognitive Walkthrough
dp	density-independent pixel
EAS	Expected Action Sequence
EvaHelper	Evaluation Helper
HCI	Human Computer Interaction
hdpi	high dots per inch
HE	Heuristic Evaluation
HIG	Human Interface Guidelines
HTML	Hyper Text Markup Language
HUI	Handheld User Interface
IBM	International Business Machines
Inc	Incorporated
iOS	iPhone Operating System
ISO	International Organization for Standardization
ldpi	low dots per inch
LG	Lucky Gold star
mdpi	medium dots per inch

MEM	Modern Evaluation Methods
MIDP	Mobile Information Development Profile
MMS	Multimedia Message Service
muEvaluationFramework	mobile usability Evaluation Framework
OS	Operating System
PSP	Platform Specific Property
RE	Regular Expression
Ref	Reference
RIM	Research in Motion
RUEM	Remote Usability Evaluation Method
SDK	System Development Kit
SMS	Short Message Service
SQLite	Standard Query Lite
TEM	Traditional Evaluation Method
THM	Think-Aloud Method
TV	Television
UEM	Usability Evaluation Methods
UI	User Interface
UIM	Usability Inspection Method
UML	Unified Modeling Language
UT	User Testing
UTM	Usability Testing Methods
WAP	Wireless Application Profile
xhdpi	extra-high dots per inch
xxhdpi	extra-extra-high dots per inch
xxxhdpi	extra-extra-extra-high dots per inch

Abstract

Mobile computing is a type of wireless communication mechanism in order to accomplish our daily activities. Mobile devices companies are increasing their mobile functionalities, screen size, and features from time to time. Android is one type of software stack which used to develop smart phone, tablet, and wrist watch applications. This is an open source platform that provides free documentations, tutorials, and applications. Therefore, mobile companies and application users are increasing from time to time. However, the usability of the applications is in question. Especially, Android user interface component usability is a major issue in mobile application development. To minimize usability problems, Google provides guidelines that can help to develop appropriate and user attractive user interface. However, the developers didn't apply these guidelines during application development because most user interface design guidelines are available in unorganized form and searching for them is time consuming task. Hence, most developer designed or developed an application based on their experience rather than using the guidelines. To solve the above problems, we designed and implemented a prototype Android application user interface evaluation framework based on the Android guidelines. Currently, there are various types of Usability Evaluation Method (UEM) available with different properties .After reviewed all the methods; we selected Automated Inspection Method (AIM) for this research work. UML design Maker tool used to draw the framework and Android studio for implement the prototype. Specifically, the study considers five user interface evaluation framework and one mobile device evaluation framework. The study includes: such as mobile device evaluation, header and footer evaluation, button typography evaluation, line length evaluation, margin evaluation and action bar menu item evaluation frameworks. Then, the proposed frameworks are integrated with different user interfaces and evaluated on various mobile devices. The proposed prototype evaluation results are compared with their guidelines values. According to the evaluation result, we concluded that the proposed framework can evaluate the evaluated user interface based on the guidelines.

Keywords: User Interface (UI), Usability Evaluation Method (UEM), Automated Usability Evaluation Method (AUEM), Automated Inspection Method (AIM), Design Guidelines.

CHAPTER ONE

1. Introduction

1.1. Background

Mobile is one of the different types of wireless communication mechanism in order to accomplish our daily activities. Simply, users can communicate each other without affected by the geographical location. That means, the receiver may live far away from the sender. But, they are able to communicate each other with at a time. Back (the early 90s) mobile application was used to perform a simple task like making calls, sending SMS and MMS, Where it all began in 1973 at New York [1].

After the smartphone was developed by IBM company in 1992 [2], mobile devices are providing a lot of services , including web browsing capabilities, video chat, email, games, word processing, multiprocessing, expandable memories , accepting applications , multitasking, and so on .Consequently, the mobile phone started to impact our lives by bringing opportunities like quick communication, saving time, increasing productivity, improving infrastructure, reducing human power, entertainment etc. The smart phone users increase from time to time as compared to computer users because of the various reasons. Most smartphone companies provide number of applications and the user can be accessed from their application stores [3], the mobile devices have small size, so the user can easily move them from place to place. A developer can develop various types of application with additional features in minimum effort, cost and time. In the evolution of the smartphone application, Android takes the leading in the development environment. It is one of the youngest aspiring platforms for application development and uses. This is because; Android provides the following services like good documentation, free tutorial resource, and open platform [4].

In contrast, Android applications have the capability of providing a smart user interface. However, the usability of the applications is in question. Especially, Android user interface component usability is a major issue in mobile application development. Android has guidelines for developing appropriate and user attractive user interface design, but the developers didn't apply these guidelines during application development. This leads to different inconsistency of usability in mobile application from one mobile screen to the other and from one screen density

to the other screen density. Consequently, inappropriate user interface leads to usability problems. These days, usability is becoming one of the key issues of a software product because the success and the failure of the products depend on the usability of the applications. That means, it is one of the indicators of quality attributes of the applications. Usability was defined by International Organization for Standardization (ISO) for user interface design in the following ways:

Usability can evaluate user interface design features like character height on display, basic ergonomic properties, physical devices and system usability in order encourages consistency across user interfaces and physical devices [5].

These days, the mobile application has effective and efficient hardware devices for more ability to run complex software. In contrast, it has a small size device and display interface. This leads to a challenge for developing appropriate software with good user experience [6]. This leads to the company needs to have a strategy for developing appropriate user interface based on mobile screen sizes and densities in order to develop user friendly products. Consequently, companies need to consider an automated testing and feedback tool as part of their release plan. That means the user needs to adapt the application to do their daily task. Moreover, the application should meet the provided users' satisfaction level and the level of external interference. Unfortunately, user interface design and usability guideline for mobile application are very limited and available unorganized form. Thus, companies need to assess a new way of user interface usability analysis, evaluation, and functionality.

Today, to improve mobile application usability evaluation methods, varies researches have been conducted in different ways. Such as traditional laboratory-based, commercial automated tools etc. However, the specified methods have their own limitations. For instance, traditional laboratory -based evaluation method is time consuming, costly and does not reflect real scenario [7]. On the other hand, there are different types of commercially available automated usability testing framework such as Flurry, Google Analytics, Localytics and User Metrix. But, they are limited in such a way that in order to use these frameworks to use them, developers must change the application source code manually to integrate the evaluation framework [6] [8] .This leads to an additional burden for developers. Most of these frameworks can evaluate the dynamic

aspects (user interaction) of a mobile application rather than the static (user interface) components of the mobile application.

The focus of this proposed automated user interface usability evaluation framework is to mitigate the above limitations by developing a proposed framework based on user interface design guidelines without the major source code modification of legacy Android applications.

1.2. Motivation

The main motivation of this research work is to solve the user interface component usability problems of Android-based mobile application. Currently, Android-based mobile applications are more popular than other smartphone applications because Android is a very open platform and the source code is freely available [8]. Hence, the developer can develop the application without investing big cost and time. Whereas this leads to the cheaper Android based mobile phones than other smartphone applications. Most smart phone users are used Android based mobile in the world which is compared with other smart phone application in 2016 [2]. However, the Android application has various types of user interface element usability problems, including in appropriate size, alignment, font, color composition and so on. And also, the usability of the mobile application is affected by knowledge of the user. Specifically, users are classified into two major groups such as common user and technical user. The technical users known everything in the application, but common users doesn't known the application due to different reasons like educational background or their experience. Therefore, most of users refrained to use smart phone applications. Considering this problem, we plan to design and develop the automated user interface usability evaluation framework for Android-based mobile application to detect user interface design usability problem at the early stage of application development.

1.3. Statement of the Problem

Currently, the use of smartphone application is an inevitable. This is because it is providing an exciting service, such as enabling communication through call and message; access the internet; access media; incorporates audio/video recording etc. In addition to this, smart phones contain a built-in keyboard, high-resolution camera, front side camera for video conferencing, touch screen etc. However, there are varies problems occurred while these applications are in use. To mention a few, unknown the exact location of the application found inside the

mobile; the procedure of the application; confusing the selection of application icons and the exact task of the specific application are the main issues and challenges in the Android mobile application environment.

These problems are identified as design usability problem for the specific smart applications. Similarly, users' different background and experience are also one of the challenges in given quality metrics. For instance, users' educational level and experience are also considered as a bottleneck for the use of smartphone application to its best. Moreover, the Graphical User Interface (GUI) of the application merely affects the usability of the application. Thus, usability evaluation is very important for both designers and developers in providing a valuable reflection followed by the respective correction before the application rises on the market. As a result, Android has user interface design guidelines for developing appropriate and consistent user interface according to screen size and screen densities. However, applying these design guidelines during application development is very difficult because the guidelines are not organized properly. As a result it takes time and required high financial resources. Therefore, the developers developed applications using their experience rather than follow the guideline which leads to usability problems.

To mitigate the above challenges, different researchers conducted a study concerning automated usability evaluation method on the smartphone applications; especially, on Android and iOS [4] [6] [7] [8] [9] [10] [11] [12] [13]. However, the previous studies entail major limitations. For instance, most of the authors used a limited quality metrics to evaluate Android application usability and most of the studies were emphasized on the dynamic aspects of usability evaluation of mobile applications . Some authors used to evaluate and analysis the usability of a mobile application instead of providing critiques except [6] [9]. In the other case, some studies suggest a tedious and manual way of integration of an automated usability evaluation functions in each application features [7] [10]. The evaluation method is time-consuming and tedious for large application. A mobile user interfaces analysis framework was developed for assertions analysis of a Microsoft.NET Compact Framework environment [12] [14] , but these usability evaluation method's evaluated user interface element threshold value was specified by the developer without considered mobile screen sizes and screen densities [12]. However, according to mobile guidelines [15] [16] , the same user interface element has consisting various shape, dimension, a

color composition in different Android versions, mobile screen sizes or/ and screen densities. Generally, there is no researcher that propose a framework to automate user interface usability analysis for Android mobile applications based on the given Android user interface design guidelines.

Mainly the proposed study was found a solution for the above-mentioned problem using most important user interface design metrics of user interface attributes thereby propose an automated user interface usability evaluation framework applicable for Android application by using Automated Inspection Method(AIM).Therefore, the research intended to answers the following research questions.

- What is the most important quantitative metrics that are used to evaluate Android-based mobile user interface usability?
- What is the degree of problem-solving of the proposed automated user interface design usability evaluation method?

1.4. Objectives

1.4.1. General Objective

The main objective of this study is to design and implement a prototype of an automated Android application user interface evaluation framework based on the Android guidelines.

1.4.2. Specific Objectives

Specifically, the proposed work meets the following objectives.

- Defined the potential benefits of usability testing for the Android-based mobile application.
- Examined the current usability analysis trend.
- Extracted important user interface usability analysis parameters.
- Developed automated user interface usability analysis framework for the Android mobile application.
- Implemented user interface design usability analysis framework.
- Evaluated the applicability of the proposed framework.

1.5. Methodology

In order to attain the above objectives and to answer the research questions, the following methods and techniques are followed.

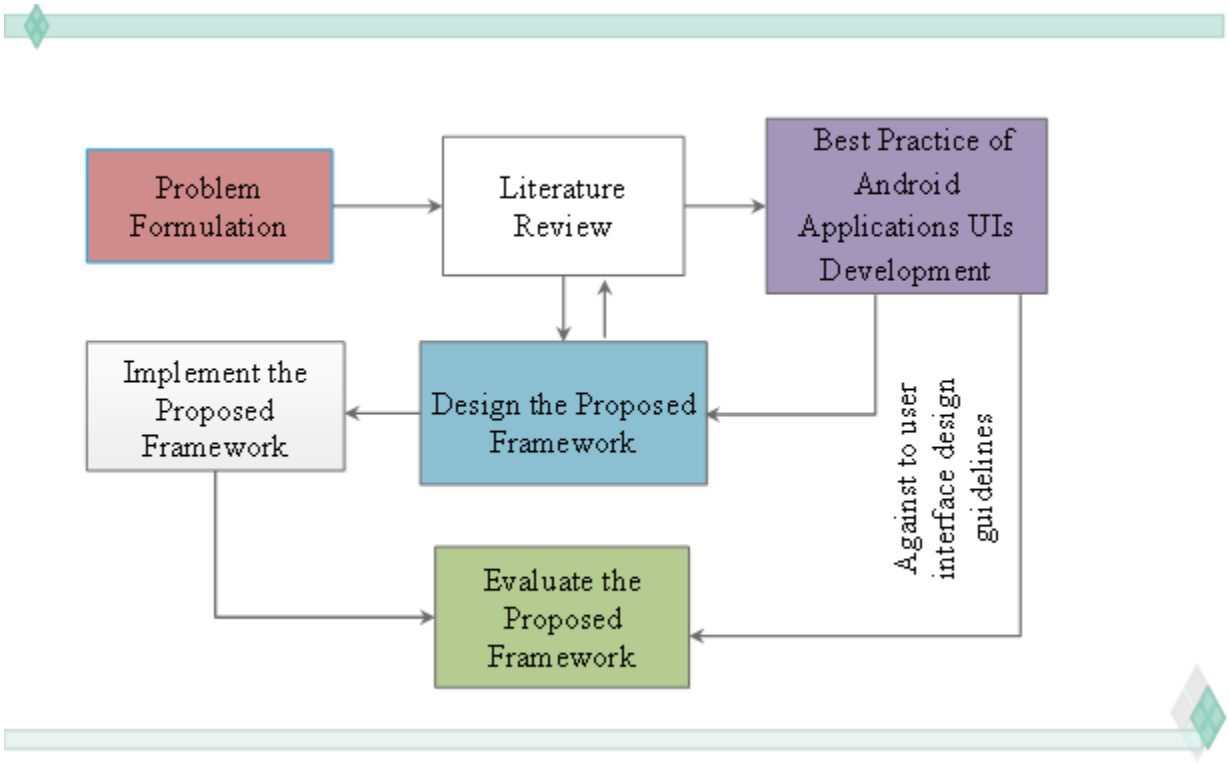


Figure 1:1The Procedures of the Research Process

We mentioned the following steps as a research methodology.

- Formulating the problem by reviewing different literature and documents.
- Collecting data from different guidelines documents and web sites related to user interface design guidelines for Android based mobile application development.
- Analyzing the current user interface design metrics and selecting the most important metrics for the Android-based mobile application development.
- Analyzing the current usability evaluation method and selecting the appropriate method for this automated user interface usability evaluation framework.
- Designing an automated user interface usability analysis framework for the Android-based smart phone application.

- Implementing the proposed framework by using open source Android development software such as Android Studio.
- Finally, evaluating and testing the functionality and usability of proposed framework against to user interface design guidelines by using an Android-based smartphones.

1.5.1. Literature Review

Several previously done related kinds of literature (research papers, magazines, conference papers, guideline manuals, and journal articles) are reviewed in order to understand the research work, to collect relevant information and to achieve the research objectives. Different techniques and tools, which are relevant to this research are analyzed, modified and used from previous works.

1.5.2. Development Tools

There are different types of designing and implementing tools that are used to design and implement the proposed framework. Such as Android which is an open source software application that is used to implement the proposed framework ; Android-based smartphone devices and emulators which are used to evaluate the functionality of the proposed framework and other designing and implementation tools are used.

1.6. Scope and Limitations

1.6.1. The Scope of the Study

This study is conducted to examine mobile application user interface design guidelines and selected the most important user interface elements metrics. We designed and implemented a prototype for automated user interface usability evaluation framework based on design guidelines. We integrated the proposed framework with user interface and evaluated the acceptance of the framework. Finally, It can be used as the baseline for implementation of automated user interface design usability evaluation tool for Android mobile applications

1.6.2. The Limitation of the Study

The study was encountered the following limitation throughout the research process. Due to the tight schedule, the study cannot accommodate all design guidelines metrics of the user interface. The other constraints of the study there is unavailability of tools.

1.7. Application of Results

The application of the study is twofold. Software development companies get an opportunity to provide quick fixes or changes of user interface design problems before their product released. And, it also enables them to maximize the accuracy of the user interface of Android-based mobile application. Thereby user satisfaction will be increased which leads to an attractive company's profit. On the other hand, the user of the application can be shielded from giving reflection overhead and they can gain ease of use of an application. Specifically, the proposed study advantages are listed below:

- Minimizing user interface design related problems for Android-based mobile applications.
- Minimizing cost for detecting user interface related problems.
- Increasing user satisfaction.
- Detecting design usability problem at the early stage before the application released.
- Facilitating the designer and developer to improve the Android based application user interface.
- Reducing the efforts of the evaluators to make user interface usability evaluation.

1.8. Organization of the Thesis

This thesis is organized into eight chapters in the following ways.

Chapter one describes about an over view of the study such as background information, statements of the problem, motivation, objectives, scope and limitation, and application of the study.

Chapter two presents about literature review. It includes the overview of mobile application; types of smart phone operating system; mobile application usability testing; mobile device screen size and resolution, and related works.

Chapter three discuses about the research methodology which includes types of UEM, data collection method, design proposed framework method, the framework implementation method and the prototype evaluation method. This chapter also describes the design and implementation tools used for the research work.

Chapter four presents about best practice of Android application UIs Development for this research work. It consists of user interface design elements with quantitative values that are taken from user interface design guidelines and from Android development web sites.

Chapter five discuses about the proposed framework design and describe each proposed framework components.

Chapter six describes implementation of the proposed framework i.e. each sample prototype user interface evaluation component and the way of integration with Android-based user interface.

Chapter seven presents result evaluation and discussion about the framework based on sample prototype implementations.

Chapter eight consists of conclusion, recommendation and identifies future works for further research direction on this research topic.

CHAPTER TWO

2. Literature Review

2.1. Over View of Mobile Applications

Mobile is one type of wireless communication mechanism for users to accomplish their daily activities. Basically, mobile devices are divided into two major categories based on their services providing capacity, such as traditional mobile and smart phone mobile. Traditional mobile devices offer simple services like voice calling, sending SMS and MMS [1]. Whereas, smart phone devices have provides additional hardware part and application services [2]. These hardware services including high memory capacity, various types of sensors, high processor speed, large screen size and high resolution. The application services also includes web browsing capabilities, video chat, internet service, word processing, multiprocessing, accept applications, multitasking, and etc.

In this chapter we discussed about the major smart phone platform types; usability testing for mobile application; mobile device properties and examined automated usability evaluation framework for mobile application.

2.2. Types of Smart Phone Operating System

Today, there are numbers of smartphone devices available in the market. Each smartphone devices have different features based on their development platforms. The operating system is one of the key issues for developing and evaluating mobile applications [17]. Currently, there are six type of major smartphone operating system available in the world [18]. These operating systems are developed by different companies such as iOS from Apple; Windows from Microsoft; Blackberry OS from Research in Motion (RIM); Symbian OS from Nokia and web OS from Palm; and Android from Samsung. Each operating system has different features and properties because it is developed for a specific mobile device. Consequently, the usability and the functions of one smartphone device differ from the others.

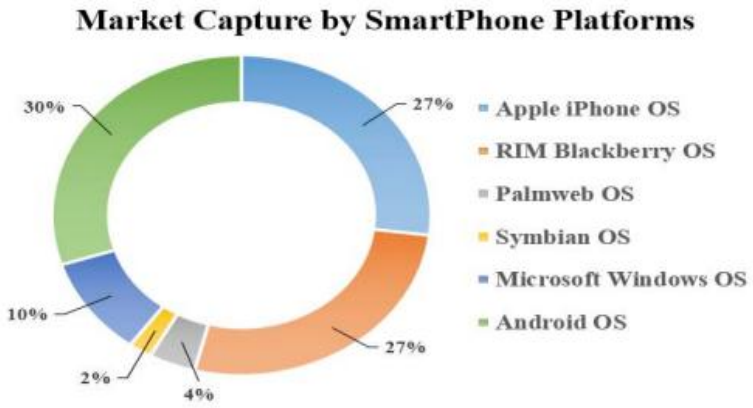


Figure 2:1 Smartphone Platforms **[19]**

The above diagram shows the six types of smartphone platforms available in the market. In the diagram, Android is the most available smartphone platform which covers 30% from the total operating systems market. Blackberry and iPhone platforms are the second rank operating systems which is covered 27% each of them. Window operating system is the third ranked platform based on the existing in the 10% market share. Web OS is the fourth rank smartphone platform that has 4% market cover. The Symbian operating system is the least available platform in the market compared to other types of platforms. It only covers 2% of the market. Each operating system is described as follows:

2.2.1. Apple iOS

iPhone operating system (iOS) is widely used types of operating system for Apple's smartphone devices which is developed by Apple Inc. Currently, this operating system is applicable for different types of Apple's products such as: iPhone, iPod Touch ,iPad, and Apple TV [18] [20]. According to [2] , it is the second most popular type of smartphone operating system in the market after Android. This strong but an expensive operating system is implemented by an objective-c programming language. iOS has the most popular direct manipulation features for easily controlling user interaction with smartphone devices by using various types of a touchscreen gesture such as tapping, swiping and pinching similar to Android platform. The iOS has numbers of versions were developed in order to improve the features of the platform, such as iOS1, iOS2 and so on. Currently, Apple provides not only the operating system; but also different types of service, like software development platform, and various types of applications. Generally, iOS has more than 300,000 applications that store on Apple's App

Store [20]. However, these applications can be downloaded on your device if and only if we paid fee. This indicates that the customer cannot get the applications for free like Android applications. This leads reducing the acceptance of the platform by the users.

2.2.2. Window OS

Window Phone is a mobile operating system. This operating system was launched by Microsoft Corporation in the name of Window Phone 7 at 2010 [2] [18].Today, various mobile manufacturing companies are developing Window Phone devices, including HTC, Nokia, Samsung, and LG. Consecutively, the Window Phone operating system was performed further update including Window Phone 8 and Window Phone 8.1 for improving the feature of the original operating system. For example, earlier Window Phone platforms were not compatible with large screen devices like Tablet and Phablets. Currently, Nokia is manufacturing 6 inch Phablets and Microsoft; so that they are developing a new operating system in order to support these large screen devices [2]. There are numbers of smartphone devices that supports Window Phone operating system such as HTC Mobile Radar, Nokia Lumia 928, Nokia Lumia 1020 and etc. One of the main backwardness of this operating system was not compatible with the earlier platforms [21].

2.2.3. Blackberry OS

Blackberry is a proprietary mobile operating system which was developed by the Canadian company RIM. This operating system was applicable only for their devices, like Blackberry mobile and Tablet devices [22]. The Blackberry operating system differs from the other types of smartphone operating system because Blackberry was developed for business usage. One of the main advantages of Blackberry device is the capability to store large numbers of email. It can supports two major protocols, such as Mobile Information Development Profile (MIDP) and Wireless Application Profile (WAP) which are used to synchronize direct a BlackBerry Enterprise Server (BES) with push-based application components including, calendar, contact, task, email, and note exchange [18]. And also, BES offered different types of service and features like security, remote wipe and capacity for Blackberry mobile device which can assess the internal network or the corporate data. Blackberry OS offers their own Blackberry Internet Service (BIS) to the customer to access their personal email, search different types of information from the internet and so on. Blackberry OS6 and O7 are types of Blackberry

operating system versions which were constructed to encourage application development. Therefore, this operating system supports an application which is implemented by different programming language such as Java is used to implement for mobile phone device and c++ or Web-based language for the Playbook Tablet devices [18].

2.2.4. Symbian OS

Symbian is a smartphone operating system which was launched by Nokia [18]. Previously, there were different hardware vendors, including Motorola, PSION, Ericsson, and Nokia were formed Symbian OS. Today, the Symbian foundations are run on Nokia only. It provides standard license agreement for the user in order to access the Symbian application. Nokia provides System Development Kits (SDKs) for Symbian application development that supports different types of programming languages, such as C++ and Java. Symbian has numbers of versions like other mobile operating systems, including Symbianˆ1, Symbianˆ2, and Symbianˆ3 and etc. These versions were realizing in a different time in order to increase the feature of the Symbian application for satisfying the end user.

2.2.5. Web OS

Web OS is a proprietary and Linux kernel-based operating system for smartphone and smart TV devices, which was developed by Palm. This operating system is called HP web OS and Palm web OS. Both a native (inside in the operating system) and third party applications are implemented by HTML, CSS, and JavaScript code. Palm was developed two types of a framework, such as Mojo and Enyo these frameworks are supporting to the developer can be easily creating web application with the same layout rules [23]. The web OS supported to display one application at a time so the user can be seen required application by moving the scroll horizontally in order to move the application from foreground to background. Applications are kept on the startup in the form of a grid structure. And also, commonly used applications can hold on the quick-launce bar. The UI supported various types of features, including touch and gesture commands swap, pinch and tap similar to Android and iOS platforms.

2.2.6. Android OS

Android is a software stack for smartphone device which includes an operating system, middleware, and key applications [17] [24]. It is a Linux-based operating system with open source license for touchscreen devices. Android was developed by an Open Handset Alliance which is a group of over 30 companies and that are governed by Google [25]. In addition to touch screen devices, Google has developed Android TV for television; and Android wear for wrist watches, with their own user interface. Currently, Android is the most widely used platform in the world. In 2016 research showed that 43% of users are used Android-based mobile applications, 40 % there is iOS users and other types of mobile users [2]. This indicates that Android is the most popular types of smartphone platform because it provides various types of services, including open source platforms; many tutorials; good documentation, and freely available source codes on the internet [4].

And also, Android offers user interface for touchscreen devices in order to control objects based on direct manipulation mechanism by using touchscreen gestures, such as tapping, swiping and pinching with a virtual keyboard. It has popular development tools like Android Studio. Therefore, the developer can develop an application in short a period of time. Android has more than 1 million applications available in Google Play Store so that users can use the applications by downloading on their devices. The performance of Android-based smartphone device is more optimized than iOS-based mobile devices because Android is java based platform and it is used Dalvik Virtual Machine (DVM). The virtual machine gives high performance to Android [20].

Currently, various smartphone device manufacturers offered a closed platform like Apple, Window Corporation. But, Google prefers an open source approach in order to develop a new application with new features and also to provide various services freely available in Google Plays Store. In iPhone platform, an application is running on a single UNIX Kernel [20]. This has a limitation, if there is a problem during an application running, and then the problem can affects the whole application. On the other hand, each Android application is running independently so any single application problem cannot affect the whole application. Consequently; now Android is the most popular smartphone platform in the world. Currently, various smartphone device manufacturers such as Google, LG, Samsung, HTC, Sony, ASUS, are using Android platform for their mobile products [20].

2.2.6.1. Android Operating System Architecture

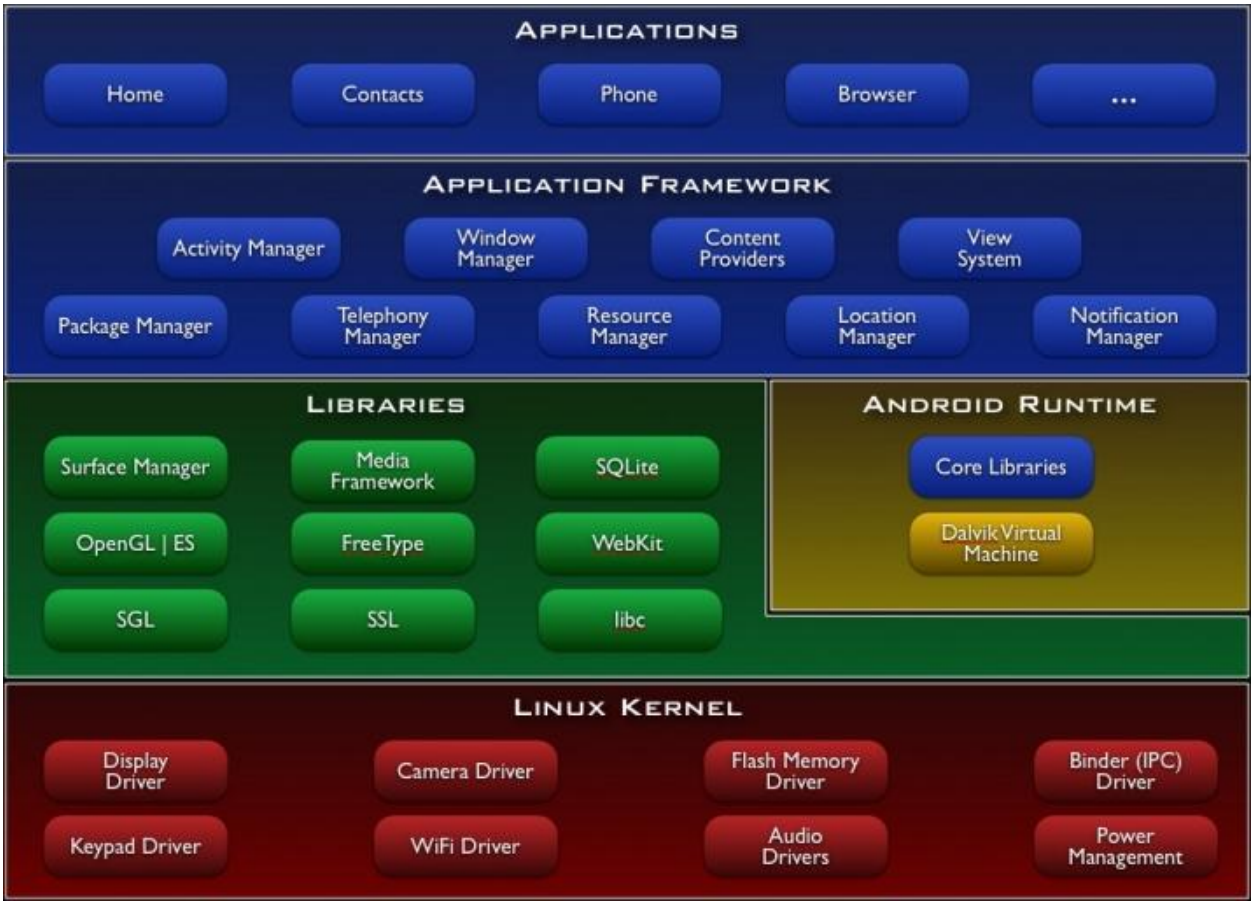


Figure 2:2 Android Operating System Frameworks **[26]**

The above diagram shows the internal architectures of Android OS. This architecture has four basic layers, such as applications, application framework, libraries and Linux kernel. Each layer also constructed from numbers of components and these components are described as follows [20]. An application is the top most layer of the Android operating system framework and consists of core applications, including contacts, SMS program, calendar, browser, maps, and so on. Android application usually developed in java programming language by using the Android Software Development Kit (SDK). It also supports other types of development tools that are called Native Development Kit (NDK) for applications or extensions in C or C++.

An application framework is the second layer after application layer which consists of various reusable components such as activity manager, package manager, telephony manager, window manager, resource manager, content providers, location manager, view system, and notification

manager. Besides, application framework has reusable components, such as libraries and services which are used during application development. This leads to saves time and effort of the developers. The third layer of the Android operating system is called library. This library classified into two major components such as Android libraries and runtime. The libraries consist of different components, including Surface Manager, Media Framework, SQLite, OpenGL/ES, SGL, Free Type, SSL, Web kit, and libc. The main function of system libraries are supporting application framework and connecting application framework with the central parts of Android is called Linux kernel [2]. They also offer service for the developer through application framework. These libraries are implemented by C or C++ programming language.

The Android Runtime also consists of two basic parts, including a Java Core Library and Dalvik Virtual Machine. It offers several core libraries in order to support the functionalities in the core libraries of java [18]. The Android virtual machine is called DVM depend on Linux kernel which enables to run Java applications. Linux kernel is the nuclear(central) parts of the Android operating system which provides core system service, such as memory management, internal protocol, process management, power management, security and so on [2] [18]. It also consists of various types of drivers in order to control hardware parts of a mobile device such as display drivers, keyboard drivers, camera drivers, Wi-Fi drivers, flash memory drivers, audio drivers, and binder (IPC) drivers. Linux kernel is developed by the high-level API is called C programming language and the application of Android is implemented by java and run with DVM. This virtual machine is used to translate the java to Dalvik dex-code at compilation time. This combination brings up various types advantages like security features, including UNIX User Identifiers (UIDs) and file permissions; improving memory utilization by using shared memory management; and preemptive multitasking. Each Android process runs on separately under unique UID with separate permission. This facilitated the application cannot read or write each other during performing any task. Therefore, the entire applications didn't affect by one application process problem. This is the main advantage of Android operating system platforms.

2.2.6.2. Android Operating System Features

Currently, smart phone companies are developing numbers of OS versions to enhance their platform features. Hence, Android has numbers of consecutive versions. The latest version has both the previous and additional features without major changes to the whole system. According to [20], Android versions are described as follows:

The first version of an Android application is known as Cupcake and Donut. This version has consisted of the following major features, such as display notification in drop-down form; widgets are used to access the home screen function without open the applications; a Gmail is integrated with Android application ; it provided application download from Google's play store; CDMA supports from different providers; the user can access the keyboard on the touchscreen; video can be captured and uploaded to YouTube, and it includes the third application development kit and support. The second version of Android is called Android 2.0. This version included the following types of application software such as Eclair, Froyo, and Gingerbread. Besides, they are added to the following major functionalities into the previous version; including multiple Google account; Google map application; quick contact search abilities; transferred from speech to text; it has five home screens; enhanced gallery; PIN lock ability; and front face camera support.

The third type of application software version is called Android 3.X and also known as Honeycomb. It supports various types of additional features such as action bar and application can be access without physical button. This OS version also provides two mobile devices can be connected each other by touching share files is called Android Beam.

The fourth version of the Android software stack is called Android 4.3 which is also known as Jelly. It supports running different types of devices which consist of low-level energy smart accessories like heart beat monitoring, thermometers, pedometers and so on. It also provides auto-complete functions in the dialed-up suggesting content name or phone numbers during used holding the start key; the notification bar consists of additional function like the user can control and interact with the status bar notification.

Table 2-1Android Operating System Versions [27]

Code name	***Version number***	***Initial release date***	***API level***
Alpha	1.0	September 23, 2008	1
Beta	1.1	February 9, 2009	2
Cupcake	1.5	April 27, 2009	3
Donut	1.6	September 15, 2009	4
Eclair	2.0 – 2.1	October 26, 2009	5-7
Froyo	2.2 – 2.2.3	May 20, 2010	8
Gingerbread	2.3 – 2.3.7	December 6, 2010	9-10
Honeycomb	3.0 – 3.2.6	February 22, 2011	11-13
Ice Cream Sandwich	4.0 – 4.0.4	October 18, 2011	14-15
Jelly Bean	4.1 – 4.3.1	July 9, 2012	16-18
Kit Kat	4.4 – 4.4.4	October 31, 2013	19
Lollipop	5.0 – 5.1.1	November 12, 2014	21-22
Marshmallow	6.0 – 6.0.1	October 5, 2015	23
Nougat	7.0 – 7.1.2	August 22, 2016	24-25

Android version 4.4 up to 7.1.2 described based on [27] in the following ways:

Android 4.4 is grouped under fourth version Android OS which is called KitKat .It provides refresh interface with white component; restrict application during access the external storage; wireless printing technology and loading Google+ photos capabilities. The fifth version of the Android OS is called Android 5.X which is also known as Lollipop. Lollipop has a code name is known as “Android L”. It supports Material design, improvement notification (refresh tray and quick settings); Lock screen offers shortcuts and notification settings, Audio input and output via USB devices. The sixth version of Android OS is called Android 6.0 which is also known as Marshmallow. Marshmallow has a code name is called "Android M". It provides “Doze” power saving, App standby feature, native fingerprint, and automatic data backup and restore for apps. The seventh Android OS is called Android 7.X which known as Nougat. It provides split-screen display mode, inline replies to notifications and automatic system updates.

2.3. Mobile Application Usability Testing

Usability has been defined in software quality standards in different ways. According to ISO/IEC 9126 standard [25], usability is a type of quality attributes for software products which have the following major characteristics, including understandability, learnability, and operability. In ISO-9241-11 standard, usability described in terms of the following three major attributes, such as effectiveness, efficiency, and satisfaction based on a specific context of use [28]. Based on ISO 9241-210:2009 software quality standard, usability defined as "the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use" [5]. According to ISO/ IEC 25010:2011, usability is an easily use a user interface and measuring of user's experience about the quality of a system, a product or a service and the user can accomplish their task successfully in the given time with satisfied by their experience [5].

The various types of definition indicate that usability is a concept rather than concrete which means that is not a single point of measurement for all services or products [29]. According to [30], the developer should focus on usability and user experience in order to develop a successful and high-quality application. Therefore, user interface design usability evaluation is an important issue for computer and mobile application for captured usability related problems at an early stage. As the result, the developer is able to develop user attractive and easily understandable applications. When the application is attractive, the user can perform their task without major challenges. However, most companies are focused on the features of the product rather than the usability of the system [31]. This shows that the companies have given more emphasis for functional testing rather than usability testing. Usability testing should be performed for any application since the success or the failure of an application is based on user satisfaction. Hence Software Development Companies should apply user-centered design and performing usability testing for the application.

However, the novelty of mobile application and unique feature of a mobile device become the main challenge for conducting usability measurement activity on a mobile device [30]. Some of the main challenges are: limited bandwidth; limited memory size; wireless connectivity; small screen size; different resolution; context-wares (geographical location) and limited processor power. Hence, usability evaluation mechanism of the mobile device is differed from the desktop

computer. Currently, usability has numbers of attributes and metrics which is used to conduct usability evaluation of a product. However, the evaluator should be identified usability attributes and metrics before conducting an evaluation process because of it is difficult to conduct all attributes in terms time, cost, and human power.

Hence, there are different types usability models described with various types of usability problems. Keenan, S. L. and et.al [32], proposed Usability Problems Taxonomy (UPT) model for classified usability problem into a hierarchal structure. This classification helps the software designer and developer for deeply understand, capture and detect usability problems. Based on, *Figure 2:3* UPT consists of two major components of a parent hierarchy such as an artifact component and a task component. The artifact components focus on the problems of individual user interface objects that effect on the user during interaction with the user interface. This includes inspect or looks user interface elements (visualness); understanding of words (language); or control user interface elements (manipulation). Whereas, task component categories focus on unexpected problems related to user moves on the user interface during performing their task. Basically, task classifications described the problems related to the structure of the system (task- mapping) and the system skill to give guidelines to the user in order to perform their task in an appropriate way (task-facilitation).

This shows that usability problems have various types of properties. Consequently, different researchers were studied and constructed numbers of usability evaluation framework for collecting and analyzing usability data. But, it requires further study on usability evaluation of mobile application especially, a graphical user interface component in order to minimize usability problems before a product is realizing into the market. Basic mobile user interface elements are icons, Widgets, resources (images), menu, and action bar, dialog box and so on. The properties of these elements will be more elaborated in the *chapter four*.

Generally, there are different types of usability evaluation method available in the mobile application development, such as inspection method, test method, remote usability evaluation method and automated usability method. These methods will be described clearly with more explanation in the next chapter. This classification is described after the following UPT diagram:

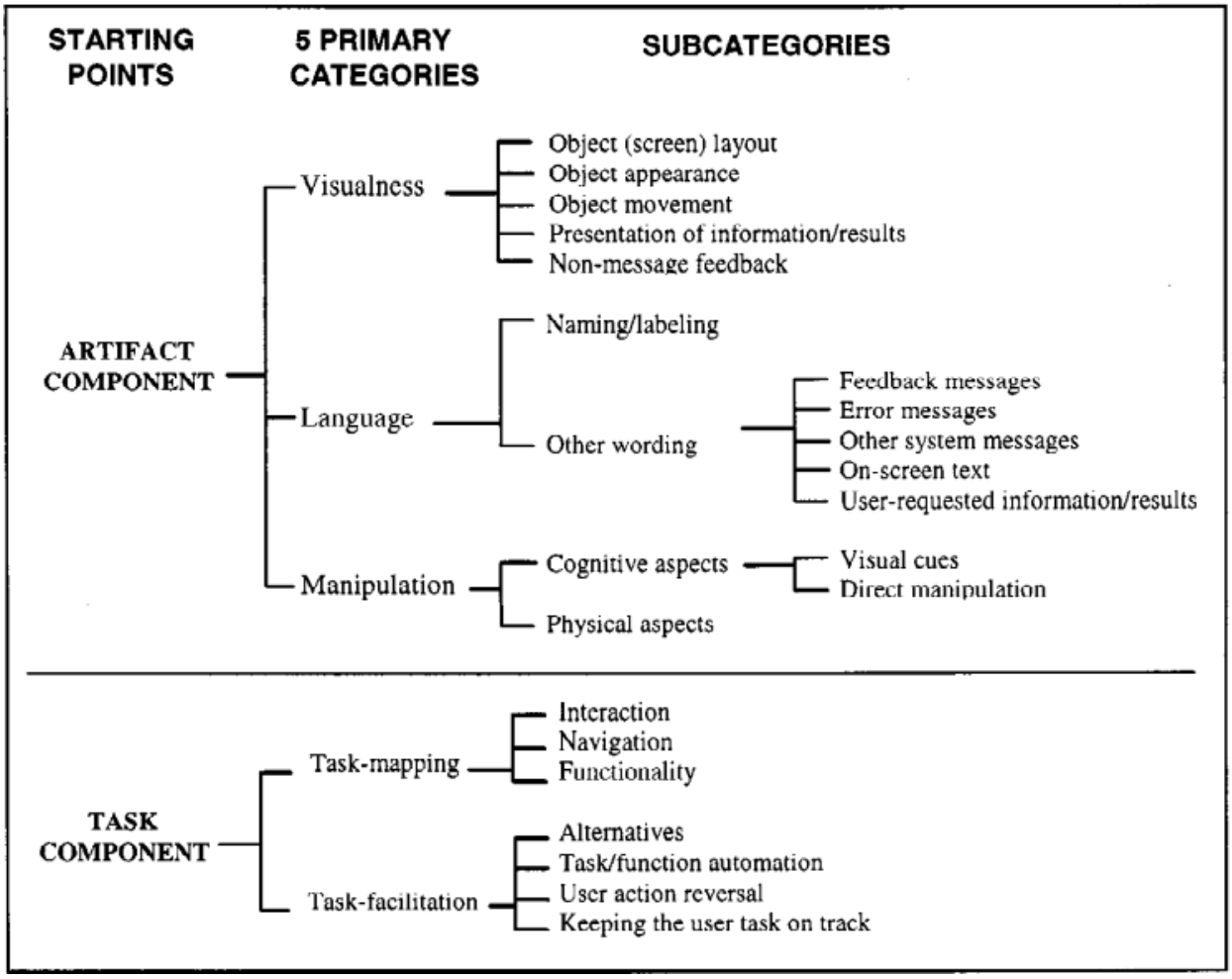


Figure 2:3 The Usability Problem Taxonomy (UPT) **[32]**

2.4. Mobile Device Screen Size and Resolution

Android provides to develop verities type of mobile applications that support multiple screen sizes and densities. Before describing mobile screen size and density, we defined important terms and terminologies about the mobile screen. These terminologies are described in the following ways with their mathematical formulas. These equations are used the following abbreviations such as h for mobile height in pixel, w for mobile width in pixel, and screen size in inch. According to [33], Android mobile device has the following terms and concepts.

- ***Screen size*:** it represents the actual physical screen size of the mobile phone. It is the diagonal measurement of a mobile phone which is measured by inch. The screen size of a physical mobile phone is calculated as:

$$screen\ size = \sqrt{(\frac{h}{dpi})^2 + (\frac{w}{dpi})^2}$$ ---------- (1)

Equation 2:1 Calculate Screen Size

- ***Screen density:*** it defines the numbers of pixels in the given actual screen, usually defined by dpi (dots per inch). Screen size and screen density are attributes of mobile phone which are used to develop the appropriate user interface.
- ***Pixel (px):*** it is the smallest element of an image (digital display device). It is made up of a composition of red, green and blue colors.

$$\text{px} = \text{dp} * \left(\frac{\text{dpi}}{160}\right)$$ ---------- (2)

Equation 2:2 Calculate Numbers of Pixels in the Screen

- ***Pixel per inch (ppi):*** is the numbers of pixels per inch image, ppi affects' on the resolution of mobile screen. This means, when the number of pixels per inch high, the size of the image (printout) decrease but a quality of an image increase. To calculate ppi by using the following formula when giving a screen the screen resolution and the screen size.

$$ppi = \frac{\sqrt{w^2+h^2}}{screen\ size}$$ ----------(3)

Equation 2:3. Calculate Numbers of Pixels per Inch

- ***Dot per inch (dpi):*** is the number of dots (pixels) in a given physical screen. Most of the time, dpi is used for printer device whereas ppi for screens. However, in Android device, both have the same meaning.
- ***Density-independent pixel (dp):*** is a virtual pixel unit which used to express layout position or size in density independent way. The px can be converted into screen dp by using the following of formula.

$$\text{dp} = \text{px} / \left(\frac{\text{dpi}}{160}\right)$$ ----------(4)

Equation 2:4 Calculate Density Independent Pixels

- ***Scalable pixels (sp):*** In Android application development, sp is similar to dp except sp is used to measure font size in density independent way.
- ***Screen Resolution:*** the total numbers of pixels in screen that it represents the numbers of a pixel in x direction and numbers of pixels in the y direction. Basically, the screen

resolution hasn't directed affect an application support of multiple screens. Therefore, multi support adaptation is affected by screen sizes and densities of a mobile phone.

- ***Orientation:*** Android has two types of orientation from user's point of views such as landscape and portrait. That means the aspect ratio is either wide or tall respectively. Different devices have a different operation of orientation in default and the rotation change during the run time

To simplify and design an appropriate user interface, Android mobile devices are categorized into four screen size and six screen density classes. These are illustrated in the tables below.

Table 2-2 Types of Mobile Based on Screen Size **[15]**

Mobile Screen	***Screen Size (inches)***
Small	2-3
Normal	3-5
Large	5-7
X-Large	7-10(only for tablets)

The above table illustrated the classification of Android mobile device physical size based on mobile screen parameters.

Table 2-3 Screen Densities in Dpi **[33]**

Screen Density	***Value (dpi)***
low dots per inch (ldpi)	~120
medium dots per inch (mdpi)	~160
high dots per inch (hdpi)	~240
extra-high dots per inch (xhdpi)	~320
extra-extra-high dots per inch (xxhdpi)	~480
extra-extra-extra-high dots per inch (xxxhdpi)	~640

The next diagram shows that the combination of actual screen size with its inch value and screen density values with its generalized density.

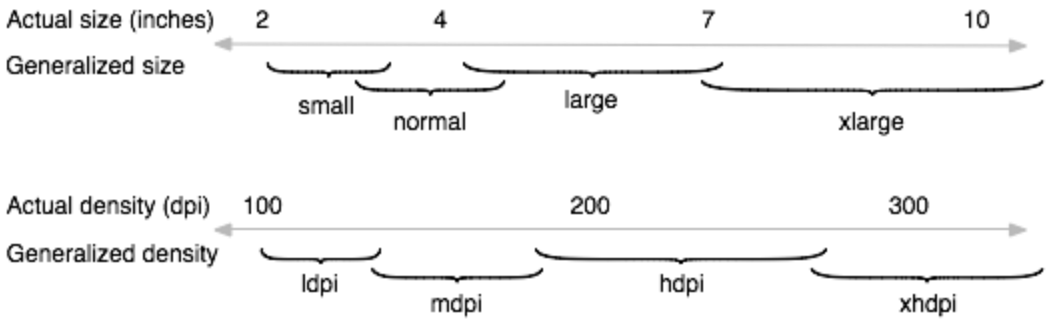


Figure 2:4 The Physical Sizes and Screen Densities Mapped into Generalized Form **[15] [33]**.

The name of the density referred to as a mount of pixels in the physical screen. For example, low screen density means fewer numbers of pixels in the physical mobile screen than normal and high screen density mobile. That means if the same screen size may have different screen densities. This indicates that screen size is independent of screen resolution.

2.5. Related Works

This section described the existing automated usability evaluation framework for a mobile device which is done by different researchers in different time.

Baker, Simon, et al. [12] developed usability evaluation framework for applications which are developed by Microsoft.NET Compact Framework environment. It is called *Handheld User Interface (HUI) analyzer tool*. The analyzer tool has been developed for capturing, analyzing and reporting usability defects. Basically, the HUI analyzer performed three major types of user interface analysis, including comparison checking, assertion checking, and hotspot analysis. Comparison checking compared an Expected Action Sequence (EAS) with Expected Action Sequence (EAS) of a specific task. Assertion checking usability evaluation results are either true or false based on predefined threshold. This assertion checking is used to evaluate font size, clutter (free space), and color usability problems by using explicitly metrics. Action sequence assertions are counting mouse click, numbers of the input text, and amount of resizing user interface components. Hotspot analysis presented user interface components by using color map descriptive approach similar to [9]. Generally, The HUI analyzer offered the above usability evaluation attributes with extensibility, and process model independence. However, it had the following limitation. First, the degree of usability problem detection is based on the quality of EAS description information. Second, the developer always recorded EAS of each particular

task. The qualities of assertions checking are based on the degree of specified thresholds because it may lead to false positive results.

Similarly, Fiora T. W. Au et al. [14] proposed a user interface evaluation tool which is called *automated Handheld User Interface Analysis (HUIA*) *testing framework* for mobile applications. This HUIA testing tool evaluated similar usability attributes with HUI analyzer tool [12]. However, the HUIA was different in two major activities. One, usability issue was identified by using case studies on three different types of user interface. Second, the *HUIA* was compared with different usability analyzer tools based on their functional requirements. Whereas, the HUI analyzer tool was compared with user testing by using the same usability attributes in both evaluation mechanism.

Daniel Bader [9] designed and implemented automated remote usability evaluation framework for Apple iOS mobile applications which is called muEvaluationFramework. This framework has different types of software based sensor to collect and display usability data. Basically, the muEvaluationFramework supported different types of usability source data such as "User-input events", "User-interface events", "Device sensor events ","Application events ", and "Custom events". Additionally, the framework can detect user interface element usability problem when violated a quantifiable Human Interface Guidelines (HIG) of iOS mobile application [34]. It has three consecutive phases such as capture, analysis, and critique activities. Finally, the muEvaluationFramework generates summarized evaluation reports. However, the muEvaluationFramework has a limitation. It detected only "Display a hint in the text field if it helps users understand its purpose" and "Maintain a hit target area of at least 44 x 44 pixels for each toolbar item design problems. This framework required network infrastructures.

Some researchers are proposed automated usability evaluation tool and framework which used to event logging framework. such as *Automated Usability Testing for Mobile Applications(AUTMA)* [6] , *A toolkit for usability testing of mobile applications* [7] , And *An EvaHelper (Evaluation Helper)* [10]. These evaluation frameworks have common properties .including all are evaluated a dynamic aspects of user interfaces, they are used consecutive activates. However, AUTMA is automated remote usability evaluation tool for iOS mobile application. And also, this framework offered automated critique for detected usability problems similar to muEvaluationFramework [9]. But, this automated usability evaluation activities are

performed without the help of a developer except an integration of the tool with the host application and data can collect without using sensors. Usability problems are a sequence of interaction design patterns. Including, “User misses UI-Element several times”; “User repeatedly touches UI elements without functionality”; “User switches back and forth between two views multiple times” and “User touches the screen accidentally and activates view change” usability problems. For each, usability problems have HCI design pattern solution. These patterns are stored in the form of pattern recognition based on Regular Expression (RE). This usability evaluation tool has four consecutive processes such as preparation, capture, analysis, and critique phases. Each user interactions are represented by the finite-state-machine diagram. However, this framework can detect limited numbers user interface usability properties known as task mapping categories like interaction, navigation, and functionality of usability problems [32] and it required network infrastructures similar to muEvaluationFramework [9].

Whereas, the toolkit is usability evaluation tool which proposed for Android-based mobile application. The toolkit performed different type’s activities like capturing, packing and transmitting of UI events to the remote server like muEvaluationFramework [9]. Basically, it captured four types of UI events, including view, dialog, menu and others. However, the degree of code modification of a toolkit is based on the hierarchal organization of the application source code. If the application of the activity extended from root activity, then the developer will add one event record statement at the root activity. But, the application of the source code not organized in hieratical structure from one root activity and each event listener was implemented separately, then the developer must be added the event recording statements in each activates.

An EvaHelper (Evaluation Helper) is types of automated usability evaluation framework for Android-based mobile application which is proposed by Balagtas-Fernandez et al. [10] . It has four common consecutive processes such as preparation, collection, extraction, and analysis phases. The framework performed usability analysis and generated evaluation results by using a graph that visualizes the user interaction. Generally, the event logging source code inserted into all of the class in the application manually. And also, the framework should integrate with various types of external tools for supporting usability evaluation process. Hence, the integration task was tedious, time-consuming and additional burden on the developer.

Automated usability evaluation architecture for mobile application is called component –based architecture which is proposed by Florian Lettner and Clemens Holzmann [11] , in 2012. This proposed architecture framework presented the innovative approaches. Because, it described the way of usability and design related problems with detection mechanism based on static and dynamic usability metrics. The static metrics presented component density under single user interface screen. However, the dynamic metrics are provided by application users. Basically, the framework can use interface tree and a structure of an application to collect usability information. The interface tree consist static and dynamic metrics. Static metrics is described the structures of an application which consists of average numbers of buttons in each activity. And dynamic metrics presented the behaviors of a user during the interaction with a user interface. However, this framework described only the theoretical aspects of usability evaluation architecture without prototype implementation.

Then, Florian Lettner and Clemens Holzmann [4] applied the previous component-based architecture [11] for *sensing mobile phone interaction in the field framework*. However, the researchers implemented the framework and integrated with host application. Hence, this usability evaluation framework is different from the previous [9] [10] [11] [12] [14] frameworks and tools. This integration can perform by using dependency injection method. Finally, the authors evaluated the performance and memory consumption of the framework. This framework was better to evaluate user interaction at application granularity level like navigational path and navigational error. But, it can't evaluate the static aspects of user interface elements.

Similarly, Florian Lettner and Clemens Holzmann [8] proposed the automated usability evaluation toolkit for mobile application and integrated without major application source code modification similar to [4].however. It is unsupervised evaluation method which differs from automated remotely usability evaluation framework [9] . This usability evaluation toolkit can collect user interaction data from the entire lifecycle and identified design weakness. Besides, this toolkit has a log which used to collect user interaction data like [6]. But, the toolkit can collect and calculate usability evaluation metrics such as hit rates, hit/miss ratios, time on screen; and described statistics data which are collected during unsupervised field studies. Then these metrics are represented by graph for representing the entire interaction of the user on the

application like [6] [10]. Additionally, this toolkit also used to a navigational graph which represented aggregated, segmented or individual user interaction data. However, the toolkit can evaluate the dynamic aspect of mobile application rather than static one.

In other way, J. Šebek and K. Richta, [13] proposed automated user interface design and development framework for an Android-based mobile device by using aspect oriented approach. Basically, this framework can used to evaluate user interface component defects by using a separation of aspects like layout design, input validation, security, and data binding. Finally, an application developed by using the framework was compared with an application developed in the conventional approach. The application are developed by the framework was better than the convectional approach in terms of maintenance, code readability, and the average time it takes to launch the form and to validated the user interface. However, the framework evaluated high-level application layout design rather than detail user interface components.

Finally, there are numbers of mobile usability evaluation framework that are developed in order to detect different types of usability problems using various methodologies. However, most of these usability evaluation methods were offered automated usability analysis rather than automatically critiques except [6] [9]. Moreover, they have various types of limitations which are described above related works. Basically, in this thesis, we proposed automated user interface usability analysis framework for Android-based application based on the *application design and development guidelines* with proposed solution (critiques). The framework can be integrated with the host application without major source code modification.

CHAPTER THREE

3. Methodology

In this chapter, we described the flow of the research methodology; examining the current usability evaluation method in order to get deep understanding of our usability evaluation method; selecting appropriate user interface usability evaluation method; describing user interface element design metrics collection method; defining the design of the proposed framework method; describing proposed framework testing method , and compare and contrast available development tools in order to select appropriate tools for the proposed framework.

3.1. Usability Evaluation Methods

Usability Evaluation Methods (UEMs) are a set of techniques which used to examine software products to detect and improve usability related problems. Currently, there are various types of usability evaluation techniques available to evaluate user interfaces. These techniques have been grouped by different researchers in different ways. For example, Melody Y. Ivory and Marti A. Hearst [35] usability evaluation method classified into two major categories such as traditional and modern evaluation method. Hasan et al. [36] usability testing methods grouped into three major categories such as user-based UEMs; evaluator–based UEMs and tool-based UEMs. Generally, we illustrated all types of usability evaluation methods in Figure 3:1.

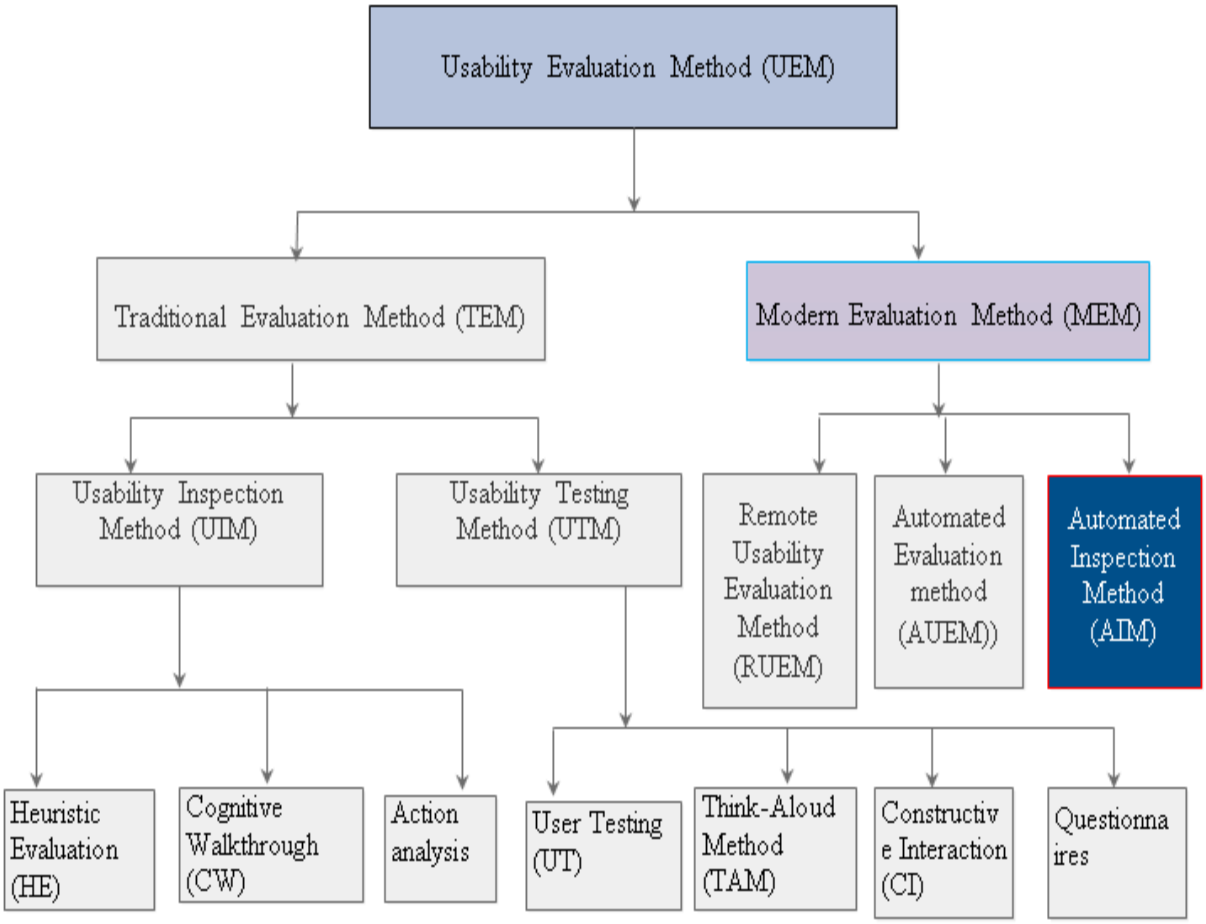


Figure 3:1Types of UEM with Structure

The above usability evaluation methods are described in the table below. This table consists of the references, description, advantage and disadvantage of each usability evaluation methods.

Table 3-1Types of UEM with Descriptions

Ref	***Description***	***Advantage***	***Disadvantage***
HE [37].	➢ To identify usability defects based on the given guidelines. ➢ Type of informal usability evaluation method. ➢ 3-5 experts should participate in evaluation	➢ Cheap ➢ rapidity ➢ Applicable for all development process phases. ➢ Applications have high recognition and	➢ It requires good experts. ➢ Difficult to know users' needs. ➢ Insufficient to detect domain-specific problems.

	session.	acceptance.	
CW [37].	➢ It is called task-oriented method. ➢ To inspect the learnability and understandability of UI.	➢ Independent from end users. ➢ Doesn't depend on an entire functional prototype. ➢ To provide early feedback. ➢ Applicable for all development process phases	➢ Tediousness and time-consuming for complex and large tasks. ➢ Emphasis on low-level details. ➢ The absence of end users.
Action analysis [37].	➢ To evaluate the set of actions when the user performs to their task. ➢ It classified into two formal and back-of-the-envelope action analyses. ➢ It performed by breaking the task into individual action.	➢ To get accurate prediction time taken a task. ➢ To get deep insight users' behavior.	➢ The representation of a task analysis is complex. ➢ It is very time-consuming. ➢ It requires high expertise.
UT [37]	➢ It is called field observation method. ➢ It focuses on usability catastrophes. ➢ It used to notes, video, audio or interaction log files. ➢ It is also used software tools like Camtasia and/or Morae [38].	➢ It is simple method.	➢ Disturbance and noise can lead to collect false results.
	➢ Usability engineering	➢ Cheap	➢ A lack of success to

THA [37].	method. ➢ End users are interpreted each user interface items. ➢ Asking end-users to think out loud.	➢ Flexible ➢ Individuals use the system in practice. ➢ To get power full comments. ➢ To detect usability problem in early stage.	cause itself. ➢ Biasing by user behavior. ➢ Time-consuming.
CI [37].	➢ A group of end users are interacting with the system to perform their task in simultaneously. ➢ It is also known as co-discovery learning.	➢ The evaluator could obtain a more natural way of thinking aloud. ➢ Get more effective results.	➢ When using unnatural settings, It get think aloud method drawbacks.
Questionnaires [37].	➢ Types of qualitative techniques. ➢ The questioner should be prepared in simple, be clear and concisions form. ➢ Indirect test method.	➢ To get subjective user preferences like satisfaction, anxieties. ➢ It used to compile statistics.	➢ It reflects subjective user's response rather than the actual response of a user interface.
RUEM [9] [35]	➢ Performed by the test user in remotely. ➢ Classified into two major approaches such as synchronously and asynchronously [9].	➢ To get real user interface usability problems which cannot collect by direct observation ➢ Evaluation can perform in user familiar environment	➢ It requires high cost and time consuming. ➢ The evaluator should require computer skill.

AUEM [9] [35]	➢ Performed by developer in locally. ➢ It classified in to four such as no automation, automatic capture, automatic analysis and automatic critiques.	➢ Reducing cost. ➢ Reducing evaluator expertise. ➢ Predicating time and error cost. ➢ Providing comparisons between multiple design alternatives.	➢ To evaluate based developer or design opinions. ➢ The absence of end user feedback.
AIM [35]	➢ It is hybrid approach such as AUEM and UIM ➢ Automatically, checking and detecting usability problems based on the given design guidelines.	➢ To reduce human power, cost and time. ➢ To minimize biased aesthetically pleasing and applying the actual design guidelines ➢ To perform without evaluator.	➢ The absence of end user feedback.

According to the above usability evaluating methods comparison, we selected automated inspection method for this research work because the objective of this research is to propose user interface design usability evaluation framework based on Android design guidelines. Hence, AIM should fit for this automated user interface evaluation framework.

3.2. Development Tools

There are different types of tools used for this research to design and implement the proposed framework. It includes Android-based mobile application development software; UML modeling tool and other relevant tools. These tools are described as follows:

3.2.1. Design Tool

UML design maker tool is a type of software design tool which is used to design the proposed framework. This design tool is more popular than other UML software design tools like Enterprise architecture because it has various software analysis and design diagrams, such as database diagram, software diagram, constricting project management activities and etc.

3.2.2. Implementation Tool

Mobile applications are implemented by different types of application development software based on their platforms. Currently, Android studio is the type of open source development software that used to develop android-based mobile application. According to [39], Android has various features including it has specific IDE, customization option, faster response when moving from one transition to the other transition, has customization option, conducted the simulation in the concepts of a module. Generally, Android studio can update new features continuously. Hence, we selected Android studio to implement the proposed framework.

3.3. User Interface Design Data Collection Method

Literature and websites are the main sources of data for this research in order to collect relevant user interface design metrics. Such as "Android design guidelines" [15] and "Application Design and Development Guidelines" [16], and Android development websites for collecting user interface element properties. During data collection time, we emphasized on the recent information because some Android application user interfaces elements are based on Android version and mobile phone screen size. However, Android is releasing a new version from time to time. Consequently, we have collected information as can as possible using the recent information from the web and the latest version of Android user interface design guidelines.

3.4. The Framework Design Method

The automated user interface design usability evaluation framework designed for automatically analyze a user interface elements based on design guidelines and finally give critiques. Specifically, the automated usability evaluation framework design to evaluate Android-based user interface elements based on the given guidelines; identify detected problems, and then give some critiques regarding a problem.

3.5. The Framework Implementation Method

The automated user interface evaluation framework implemented based on the proposed framework design that illustrated in the design section. Then automated evaluation framework has been implemented using Android studio and integrated with the host application during the preparation phase. Basically, the automated evaluation framework should implement with minimum host application source code modification to pass evaluated user interface information to the evaluation framework.

3.6. The Framework Evaluation Method

The automated user interface usability evaluation framework integrated with a host application during the preparation phase and run using different mobile screen sizes and densities. The testing can be performed by designer or developer during user interface design and development. First, the developer should add a specific source code on the evaluated interface and integrated with the evaluation framework. Then run the application in different mobile screen size and densities. Finally, the framework generates the evaluation result with user interface design defects based on the given guidelines.

CHAPTER FOUR

4. Best Practices of Android Applications UIs Development

This chapter described the quantitative value of user interface elements according to Android application design guidelines. These quantitative values are used to design automated user interface usability evaluation framework. The proposed framework should evaluate basic user interface elements. Including header and footer design , dimension, alignment(margin) , color , style, icon design and other properties of user interface components . The Android design guidelines [15] [16] and Android websites are described user interface components must be designed based on mobile screen sizes, densities, and OS. This used to develop consistent and optimum user interface element across in multiple applications. In this chapter we described the major user interface element design issue with their quantitative values.

4.1. Mobile Device Information

Mobile devices have various types of information such as screen size, mobile density, screen width, screen height, scaling factor, and Android OS information which are discussed in *chapter two.* According to [15] user interface design guidelines, mobile device information's are affected on design appropriate user interface. Example screen height is one of mobile device information which affected on header and footer design, alignment, and etc. Smartphone devices have various types of screen densities. These screen densities are grouped into six categories which are described on *Table 2-3* Screen Densities in Dpi in chapter two. These categories provided design appropriate user interface elements like icons, buttons, and other user interface components. Android OS has number of versions available which are described in *Table 2-1*. Android application element properties are varied based on its version like tab icon design [40] and other user interface components. Hence, the developer should know mobile device information. Therefore, the proposed framework can provide basic mobile device information for design appropriate user interface.

4.2. Android Application User Interface Design Guidelines

Android has application design guidelines which are available in different documents [15] [16] and Android websites [41], [42], [43] and etc. Android application has various types of user interface elements with their design guidelines. However, we cannot cover all user interface design guidelines in this thesis. Hence, we selected the following basic user interface design elements and defined their design guidelines.

4.2.1. Header and Footer Guidelines

Android has supported header and footer layout which are found at the top and bottom of mobile screen respectively. These layouts are reusable and it should use across multiple user interfaces and for scrollable contents. Android design guidelines [16] stated that the size of the header and the footer do not succeeds from 30% of the screen height and they should be consistent in all screens.

4.2.2. Typography Guidelines

Typography is a technique of arranging type to written language readable and legible when display. Basically, Android design guidelines include user interface typography properties such as language name, typeface, style, line height, line length and etc. [16] [41].According to [41] typography defined their design guidelines in the following ways. Roboto and Noto are standard typeface for Android application development. These typefaces have various font size and weight based on types of languages illustrated in *Table 9- 4* (appendix A). Roboto has six major weights which are used in different context for user interface development, including Thin, Light, Regular, Medium, Bold and Black. Whereas Noto has also all Roboto's weight with include Demi Light. Line height is one of the most important attribute for improving proper readability and to achieve appropriate placing a text in application. This line height depends on each individual font size and weight defined in *Table 9- 2* (appendix A**).** Generally, language scripts are categorized into three major groups such as English-like, Tall and Dense. Each language categories have various types of languages defined in *Table 9- 1* **(**appendix A**)**.

4.2.2.1. Button Typography Guidelines

Android user interface element properties are varied from one user interface to the other user interface based on language script type. Hence, button design properties specified based on language script type. According to typography guidelines appropriate button design properties defined in the table below.

Table 4-1 Button Font Size based on Language Type **[41]**

Language Script Type	***Buttons***	
	Text size (sp)	***Font Style***
English and English-like	14	All caps
Tall	15	All caps
Dense	15	Bold

4.2.2.2. Line Length Guidelines

Line length is one of user interface design issue in mobile application development because it affects the readability of a text. According to [41] , user interface should be consisted ideal range text size with in a line. Because text line is too wide, the user eye hard focus on the text. In contrast, the line is very short, then the eye will have to move back to repeatedly and the reader idea will be breaking the rhythm. Additionally, more than one free line between paragraphs is unnecessary because this may lead to scrollable text content increase and reader user break an idea. Therefore, to mitigate the above problems, the guideline specified number of characters per line based on text type in table below.

Table 4-2 Numbers of Character per Line for English Language **[41]**

Text type	***Too narrow***	***Ideal range***	***Too wide***
Body text	1-30 character	31-60 characters	>60 characters
Short line	1- 15 characters	16- 30 characters	>30 characters

4.2.2.3. Color and Contrast Guidelines

The typography guidelines [41] descried recommended color-contrast combination between texts and background of user interface. Including text color shouldn't very similar or much contrast to

the background color is hard to read. These guidelines recommended the light-colored text with dark background. And also, it is defined the minimum contrast and preferred ratio like 4.5:1 minimum contrast ratio and 7: 1 is preferred ratio based on color luminance value [42]. Basically, the guideline stated that the color combinations are considered contrast ratio based on a type of typeface defined in *Table 9- 3* (appendix A), user interface component used.

4.2.3. Margin Guidelines

Margin is a space which created around the user interface elements. This space is very important for smartphone application because smartphone devices have touch sensitive screens. Hence, the developer should ensure enough space between user interface components when develop application for touchscreen devices. Basically, when user interface has enough space between elements, then it minimizes errors related to inappropriate touch user interface element. The Android application design guidelines [16], recommended to small number of widget placed on application user interface and percentage based space between user interface elements. This percentage-based alignment will arrange the widget based on the available free width and height in the container. The guidelines stated that 2% for all left and right margins and 1% for all the top and bottom margins. Additionally, the guideline also stated that the developer should uses either top or bottom margin for creating a space between components but not from both side of a component.

4.2.4. Action Bar Menu Item Design Guidelines

The action bar is types of user interface component that is found at the top of each application. However, for more simplify the application, action bar content splits into three categories and keep at the different location on application layout like main action bar, top bar and bottom bar [43]. Generally, action bar provides varies type of advantage for user interface development including giving consistent navigation; view switching mechanism and reducing clutter. Most frequently used activities should keep on main action surface. Whereas, don't frequently used actions should keep in overflow (option menu) at the bottom bar. When design user interface content in action bar, the developer should hold appropriate numbers of actions based on the following rule [43].

- The action bar can hold varies a number of menu item based on the screen width of the mobile device. The next table shows a number of menu item per the given screen size interval.

Table 4-3 Number of Menu Items in Action Bar based on Screen Width **[43]**

Screen width in dp	***Below 360***	***360-499***	***500-599***	***600 and above***
Numbers of menu items	2	3	4	5

4.2.5. Menu Bar Design Guidelines

A menu is a part of user interface component which stores a set of user actions (commands). Basically, menu provides free space screen and navigate from one activity to the other activity or from one application to the other application [15]. Android has two major types of menu such as option menu and context menu. An option menu is store primary activities (commands) that apply globally to the current activity or start another activity. Context menu is similar to right click on the computer desktop. It has a list of menu items that can be operated in a selected context. The Android design guidelines [15] defined the standard height of an option menu based on mobile screen densities in *Table 9- 5* (appendix A). But, the width should be adjusted based on the numbers of the button and the size of an application layout. The guidelines [44] specified simple menu item properties including its alignment from a screen layout is 16dp for mobile and 24dp for tablet screen devices. Additionally, the alignment and text size are defined in *Table 9- 6* (appendix A). However, menu width varies according to the length of text character**.**

4.2.6. Tab Design Guidelines

Tab bar provides to hold user interface elements by organized related contents in the form of tabs. This is found at the top part of an Android application. This important to reduce crowded user interface elements and provided to hold number of user interface contents in a single user interface. Tab provides to explore and switch between different application views. Android design guidelines [15] defined the standard tab bar dimension based on mobile screen densities. This dimension defined in *Table 9- 7* (appendix A). There are two types of tab available in the Android application such as fixed tab and scrollable tab. Tab design guideline [45], described the width of each tab in the fixed tab can be calculated by the width of a view divided by numbers of

tab in the view. However, each scalable tab width should be calculated independently from mobile screen size. The guideline defined an alignment space between view edge and tab edge should less than or equal to 16 dp. Additionally, the guidelines illustrated about fixed tab such as maximum and minimum width in *Table 9- 8*, Padding and Indicator in *Table 9- 9*,and general tab information *Table 9- 10*(Appendix A).

4.2.7. Widget design Guidelines

Widgets are necessary aspects of user interface development because they have a collection of data and function. Then, the developers have been customized the required pre-defined widget during application development. According to user interface design guideline [16], there are various type of widget available in Android user interface development. This guideline described each widget properties in the following ways:

4.2.7.1. Button Design

Button is a type of widget which provides communication among a user with an application. There are three types of button available in the Android platform that used to different purpose such as flat button, raised button and floating action button [46]. Flat button is text only button which used for dialog toolbars. Raised button is rectangular-shaped button which lifts and displays ink reaction during touch the button. Floating action button is a type of button which used to represent the major action in an application that is used for promoted actions [47]. It is a circular-shape button.

According to [47], overflow action size is determined based on the screen width of mobile devices. when the screen width is less than or equal to 460dp, then the overflow action button size is 40 x 40 dp otherwise it is 56 x56 dp and an icon size also 24x24 dp for both screen widths. The above button types are used for different context and in different location on mobile screen which described in *Table 9- 11* (appendix A).

4.2.7.2. Browser Widget Design

Browser widget used for large contents of text (for more than 4 lines of text). And, doesn't advice using more than two browser widgets in one application because it consumes huge

amount of memory space [16]. When the browser widget used, avoid other types of widgets from user interface.

4.2.7.3. Map Widget Design

The guideline [16] recommended using a single map widget on an application because it consumes a huge amount of memory space like browser widget. When set a PSP (Platform Specific Property) is true in map widget, the other types of the widget should set false in order to display information and that has appropriate scrollable behavior. The screen level widget minimizes memory space which is consumed used by the map widget.

4.2.7.4. Segment Widget Design

When using segment widget in smartphone application, the developer should be checking the following issues. Including, when segment widgets are created, the developer should be using an orientation property rather than create either Vertical Box (VBox) or Horizontal Box (HBox) container. And also, the developer must be defining the maximum number of rows in segmented widget based on types of segment design view. In Android application, segment design view is classified into two major categories such as simple segment design view (segment has less than 5 widgets) and complex segment design view (multiple nesting and having more than 5 widgets) [16].

4.2.8. Segmented UI Design

Mobile application development guidelines [16] described segmented UI design. Basically, this guideline recommended that the segmented UI should be designed in simple view hierarchy to minimize memory consumption and to reduce performance issue. Consequently, for simple segment UI, the number of row in segment can be as high as 100, but in complex segment UI, it must be as low as 20; using pagination for more than 15-20 records display on the segmented UI or set the SPS to true for Android application in order to facilitate better scrolling performance because the segment UI widget can be reuse during scroll. However, the SPS has a drawback that is a segment UI only scrollable in the form but other user interface elements are set as header or footer.

4.2.9. Orientation Support Guidelines

Android mobile has two types of orientations property such as landscape and portable view. The guideline stated that an application should have a uniform view in both orientations [16]. To achieve uniform view in both orientations, the guideline defined the following issues. Such as using percentage based margins and padding; creating skins for both orientations in all application widgets define the "OnOrientationChange" event to dynamically change when the orientation of the screen change and use form fork . These are eliminated user interface distorted between the orientations.

4.2.10. Icon Design Guidelines

Icons are graphical elements of a user interface which used to represent application launchers or user interface components. Creating unified and appropriate icons are increasing the acceptance of the product in the market. During icon design, the designer should be emphasized on it supports multiple screen sizes. This issue can be resolved by using density specific icon sets. Android has various type of icons used in application development [48]. These icons are described as follows:

4.2.10.1. Menu Icon Design

Menu icons are a graphical representation of user interface items placed in options menu [15] [49]. According to [15] [50], menu icons are classified into two based on Android OS version such as Android 2.2 from earlier and Android 2.3 up to the current version. The Android 2.2 from earlier described a standard menu icon size and based on their shape and screen densities in *Table 9- 14* (Appendix A). Additionally, the guidelines described color plate information for this Android version in *Table 9- 15* (appendix A). Menu icon design guidelines [15] [49] described version 2.3 and earlier OS version of menu icon's size, full asset and squared icon based on a screen densities. This Android OS version menu icon shape, size and color, style, and etc. are illustrated in *Table 9- 13* (appendix A).

4.2.10.2. Action Bar Icon Design

Action bar icons are graphical elements which placed on action bare in order to represent individual action items. The action bar items are supported by 3.0 and above Android OS versions [50]. This guideline described icon size, the standard dimension sizes of action bar icon with the screen densities are shown in *Table 9- 16* (appendix A).

4.2.10.3. Tab Icon Design

Tab icons are used to represent individual tab elements in a multi-tab user interface. Basically, there are different types of tab icons should be used in mobile application development based on screen densities [40]. According to [40], the tab icon design classified into two categories such as Android 1.6 from earlier and Android2.0 up to Android 2.3. The guidelines also described a tab icon design properties for Android 1.6 and earlier versions illustrated in *Table 9- 17* (appendix A). Basically, all tab icons have 32x 32 dimensions with 1 pixel safe frames. The tab icons are used to r 163|g 163|b163 or r120|g120|b120 color palette unused tab icons. The tab Icon design dimension and style and color information for Android version 2.0 up to 2.3 are descried in *Table 9- 18* and *Table 9- 19* of appendix A.

CHAPTER FIVE

5. Design of the Proposed Framework

This chapter proposed an automated user interface element usability evaluation framework based on user interface design guidelines that described in *chapter four*. Specifically, Android application component design properties are specified based on screen density, screen size, Android OS version and typography properties. The proposed automated user interface usability evaluation framework has five major components such as: home screen, host applications, user interface evaluation components, user interface evaluation frameworks and display results.

5.1. User Interface Evaluation Home Screen

Home screen is a user interface which used to call evaluated host applications or mobile device evaluation framework. Basically, mobile device evaluation component provided mobile device evaluation information. Whereas users interface evaluations are a collection of components which are used to call its evaluated host applications user interfaces.

5.2. Host Applications

Host application is a mobile user interface which evaluated by the proposed framework. Basically, the host applications are used in the proposed framework evaluation session. The host application information should send into the specific proposed framework activity class.

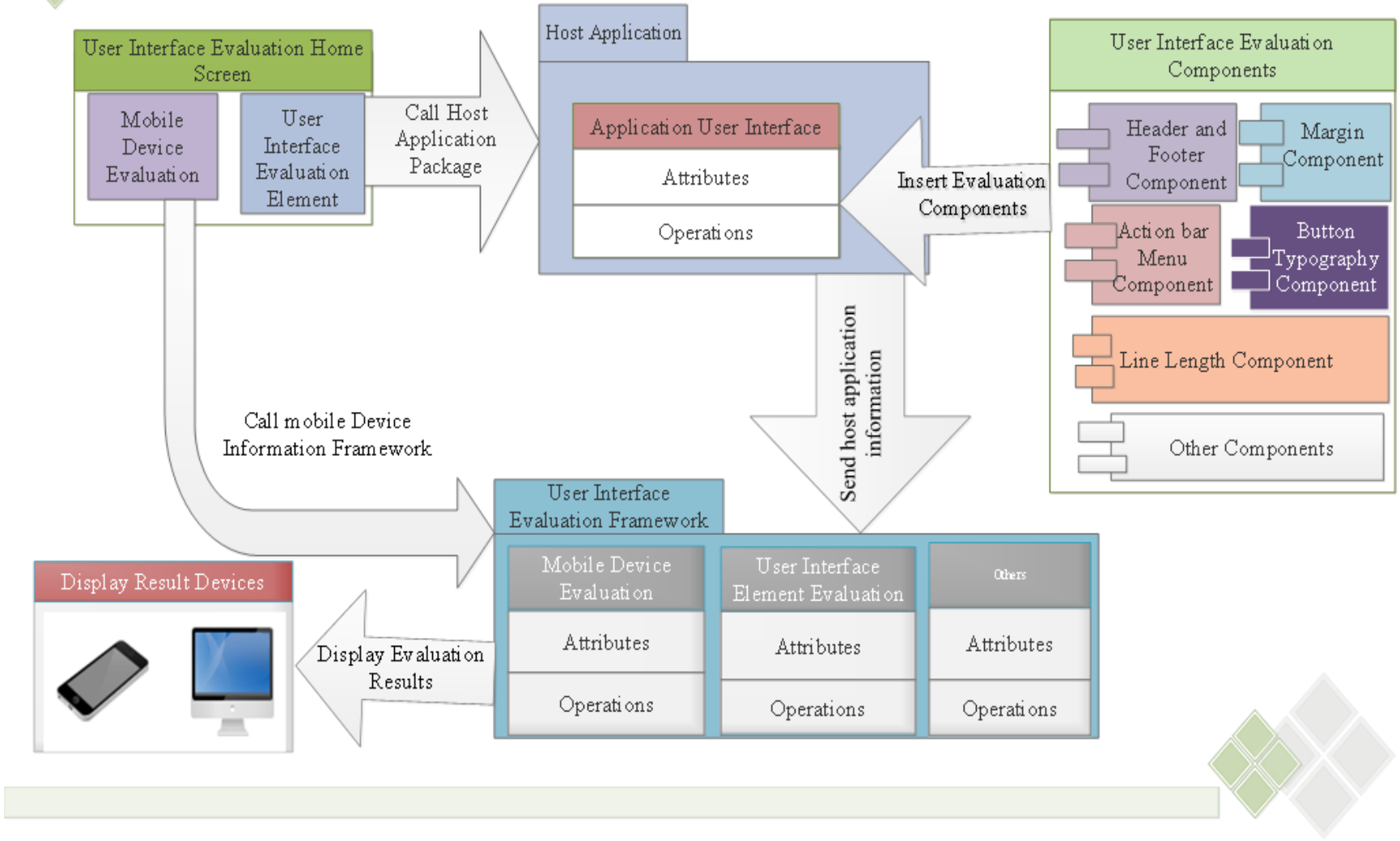


Figure 5:1 Proposed High Level User Interface Element Usability Evaluation Framework

5.3. User Interface Evaluation Components

The proposed framework has different type of interface evaluation components which are added into the evaluated user interface. The evaluation components are used to collect host application information and send to its proposed framework library class. The proposed framework has the following user interface evaluation components.

- ***Header and footer component*:** consists of header and footer evaluation attributes such as header height and footer height attributes.
- ***Button typography component*:** consists of language type, language script type, language code, text size, text style and button name attributes.
- ***Line length evaluation*:** consists of total character, start line, end line and numbers of character per line attributes.
- ***Margin evaluation component***: consists of user interface layout size and each evaluated user interface element name, margin, position, and size values.
- ***Action bar menu evaluation component*:** consist of numbers of menu items on the action bar.
- ***Other evaluation components***: are types of user interface evaluation components which are not implemented in this prototype. But, these are proposed in this framework for future implementation.

5.4. User Interface Usability Evaluation Framework

This proposed framework has consists of three basic user interface usability evaluation components such as mobile device evaluation, user interface element evaluation and others .we described each components in the following section.

5.4.1. Mobile Device Evaluation Framework

This mobile device evaluation framework consists of basic mobile device attributes such as mobile screen density, mobile screen size, and Android OS version information. Basically, this evaluation framework provided to identify user interface design defects and give critique based on mobile device. Today, there are different types of smartphone phone devices available in the world.

5.4.2. User Interface Element Evaluation Framework

The user interface element usability evaluation framework has two basic parts such as user interface element evaluation and other user interface evaluation components. User interface element evaluation component consist of currently implemented user interface evaluation components including button typography evaluation, margin evaluation, header and footer evaluation and action bar menu item evaluation. Whereas, other evaluation component consists of future implementing user interface usability evaluation components such menu evaluation component, tab design evaluation, segmented UI evaluation component, button design evaluation components and icon design evaluation components. We described each user interface element evaluation framework the following sections.

5.4.2.1. Header and Footer Evaluation Framework

The proposed header and footer evaluation framework designed to evaluate header and footer height of mobile application user interface. The proposed framework should calculate current, expected and over height of user interface. Generally, the header and footer evaluation process and calculation are shown in below.

$$Current\ Height\ = (\text{Heder Height} + \text{Footer Height}) \text{----------(5)}$$

$$Expected\ Height = \text{Screen Height} * 0.3 \text{----------(6)}$$

$$Over\ Height = (\text{Expected Height} - \text{Current Height}) \text{----------(7)}$$

Equation 5:1 Header and Footer Evaluation Formulas

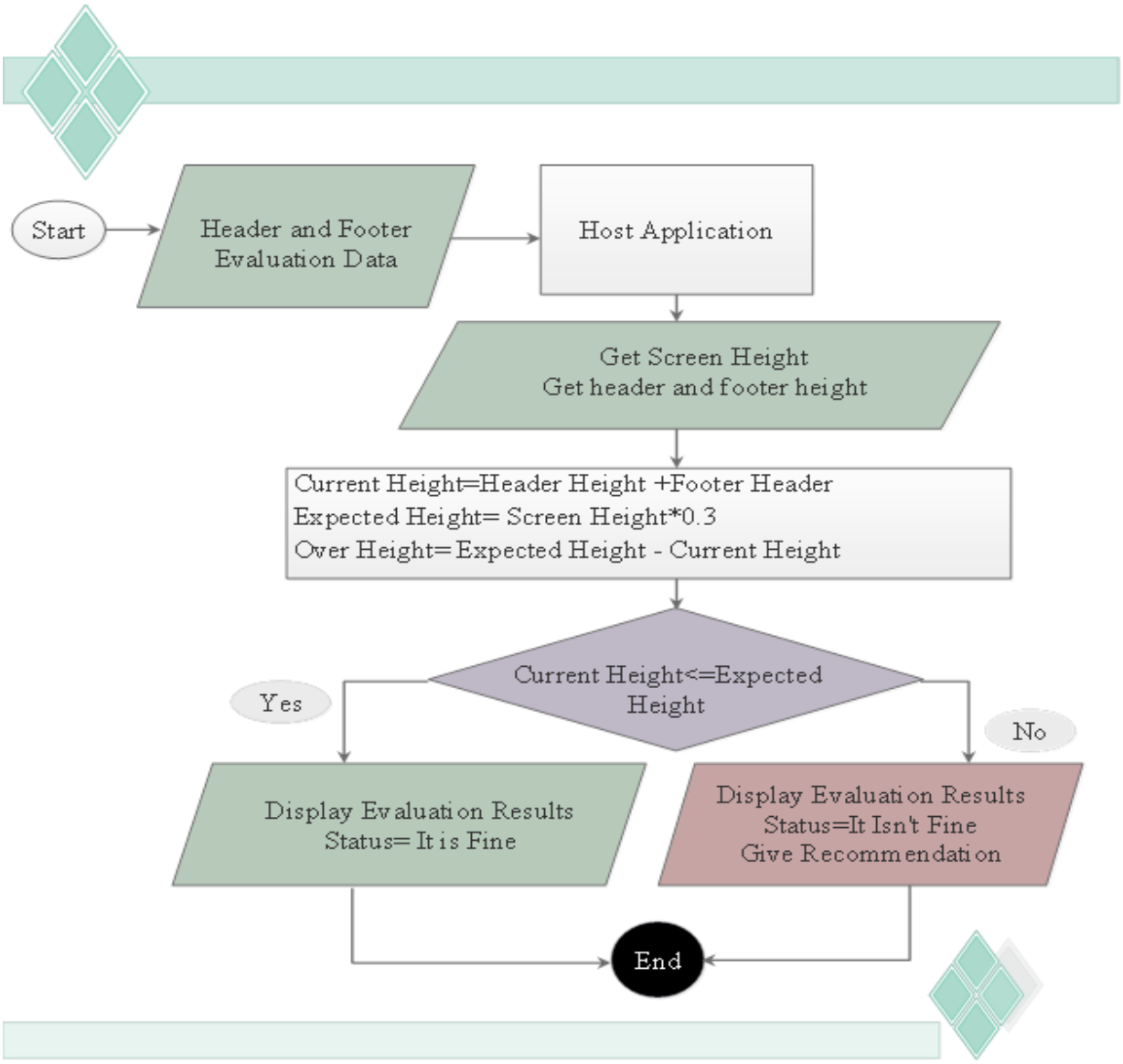


Figure 5:2 Header and Footer Evaluation Flow chart

5.4.2.2. Button Typography Evaluation Framework

This proposed framework designed to evaluate button information like language name, typeface size, font style and text effects (capital or small) based on language script type. Basically, the proposed framework evaluated button properties in multi-language. In Android, every language categorized into three groups such as English- like, Tall and Dense which are descried in the Typography Guidelines in *chapter four*. The button typography evaluation process is shown below as a flow chart.

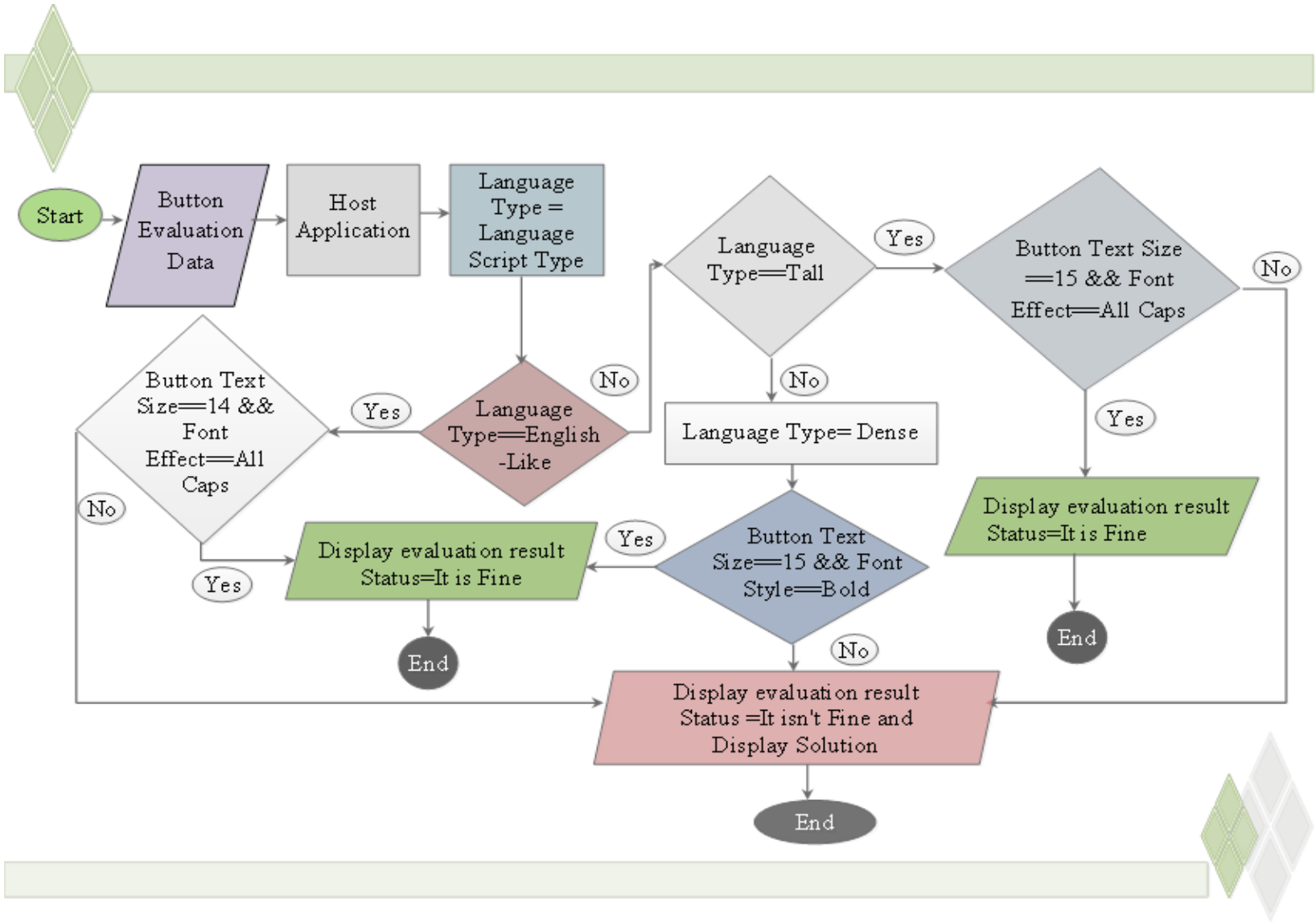


Figure 5:3Button Typography Evaluation Flow chart

5.4.2.3. Line Length Evaluation Framework

The line length evaluation framework is part of the proposed framework which evaluates the numbers of characters per line based on line type i.e. short line and text body. Basically, the proposed framework evaluate the numbers of characters per line and grouped into three categories like too narrow, ideal range and too wide. Additionally, the proposed framework can be extracted to remove unnecessary free lines.

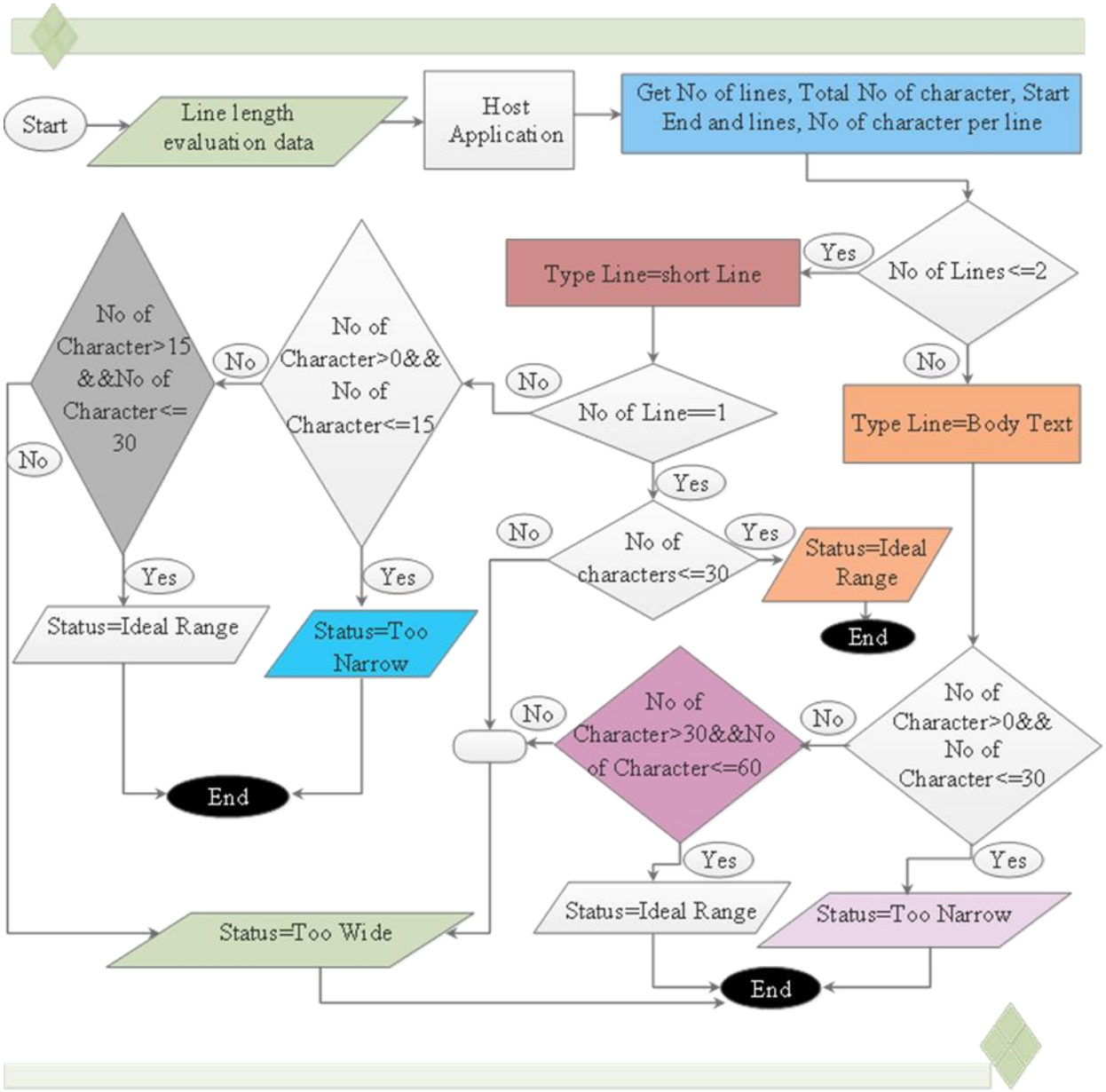


Figure 5:4Line Length Evaluation Flow chart

5.4.2.4. Margin Evaluation Framework

The proposed margin evaluation framework can evaluate the marge between user interface elements based on their container layout size. Basically, this framework collected user interface layout size and each individual evaluation element information like margin, position, and size. The margin evaluation framework calculates the total width and height of an evaluated element based on their alignment. The proposed framework also calculated free layout width and height. This framework also calculates expected free space for each user interface element position. Generally, margin evaluation framework has the following formulas.

- When all user interface elements arranged in vertically.

  $Total\ Height = (H1 + H2 + H3 \ldots + Hn)$---------- (8)

  $Total\ Widt = \frac{W1+W2+W2+\cdots+Wn}{Number\ of\ rows}$---------- (9)

- When all user interface element arranged horizontally.

  $Total\ Height = \frac{H1+H2+H2\ldots+Hn}{Number\ of\ columns}$---------- (10)

  $Total\ Width = (W1 + W2 + W3 + \cdots + Wn)$---------- ---------- (11)

- When user interface elements are arranged both vertically and horizontally.

  $Total\ Height = \frac{(H1+H2+H3+\cdots+Hn)}{Number\ of\ Columns}$---------- (12)

  $Total\ Width = \frac{(W1+W2+W3+\cdots+Wn)}{Number\ of\ \ rows}$---------- (13)

- Calculated free layout space height and width.

  $Free\ Layput\ Height = (\ Layout\ Height - Total\ Height)$---------- (14)

  $Free\ Layout\ \ Width = (Layout\ Width - Total\ Width)$---------- (15)

- Calculated expected free space left, top, right and bottom position.

  $Expected\ Left\ and\ Right\ \ Free\ Space = (Free\ Layout\ Height * 0.02)$------- (16)

  $Expected\ Top\ and\ Bottom\ \ Free\ Space = (Free\ Layout\ Width * 0.01)$--------- (17)

Where H: Height, W: Width

Equation 5:2Margin Evaluation Formulas

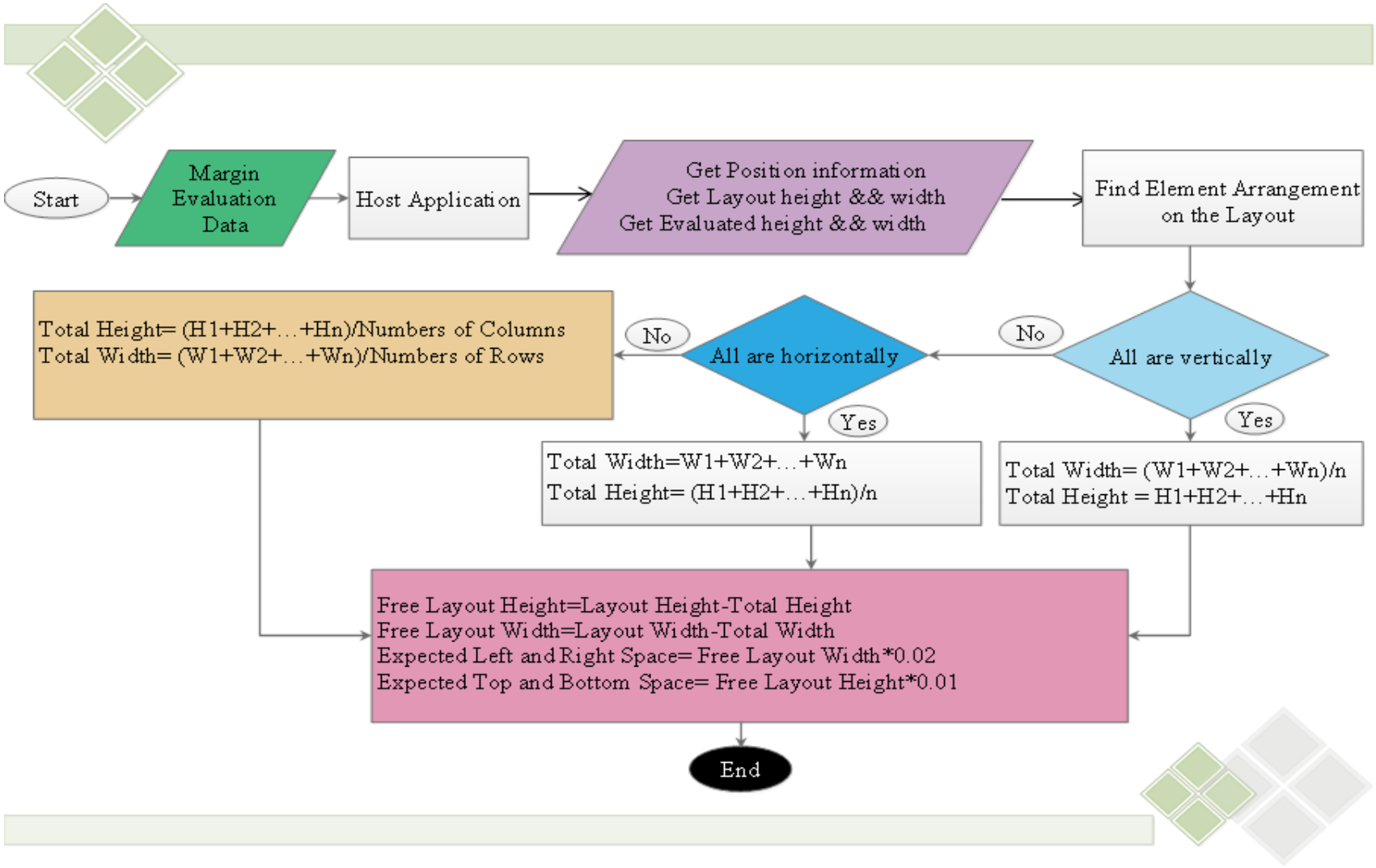


Figure 5:5Calculate Layout and Expected Free Space Flow Chart

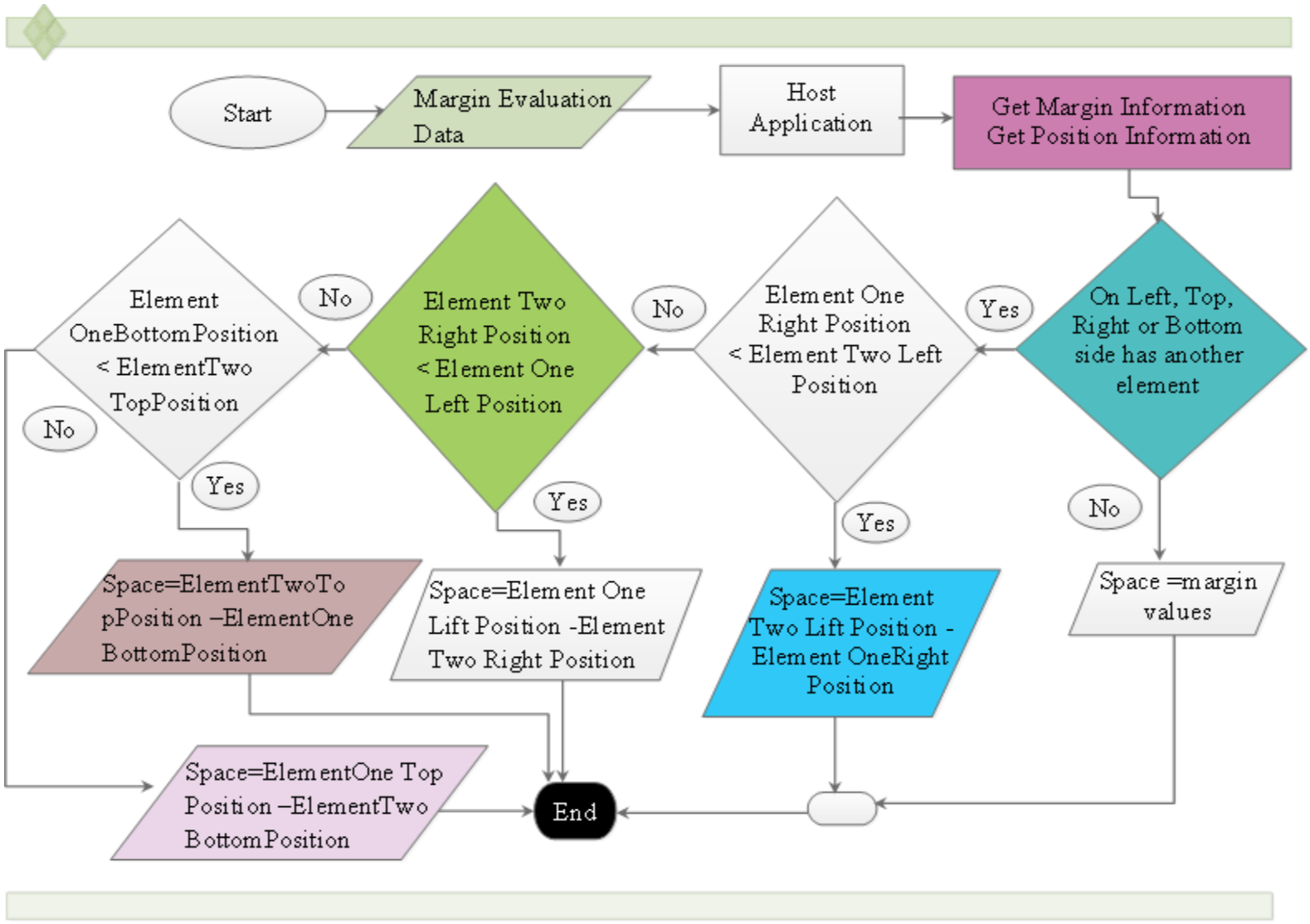


Figure 5:6 Calculate Space between Host Application Element Flow Chart

5.4.2.5. Action bar Menu Item Evaluation Framework

The action bar menu evaluation can evaluate the number of menu item on the action bar based on their mobile width in dp. The proposed evaluation framework has basic evaluation components like get mobile screen width; calculate number of menu items available in the action bar. This framework should evaluate the numbers of menu items available on the action bar based on mobile screen width interval and generate evaluation results with recommendation. The proposed action bar menu item evaluation framework process is shown in the flow chart.

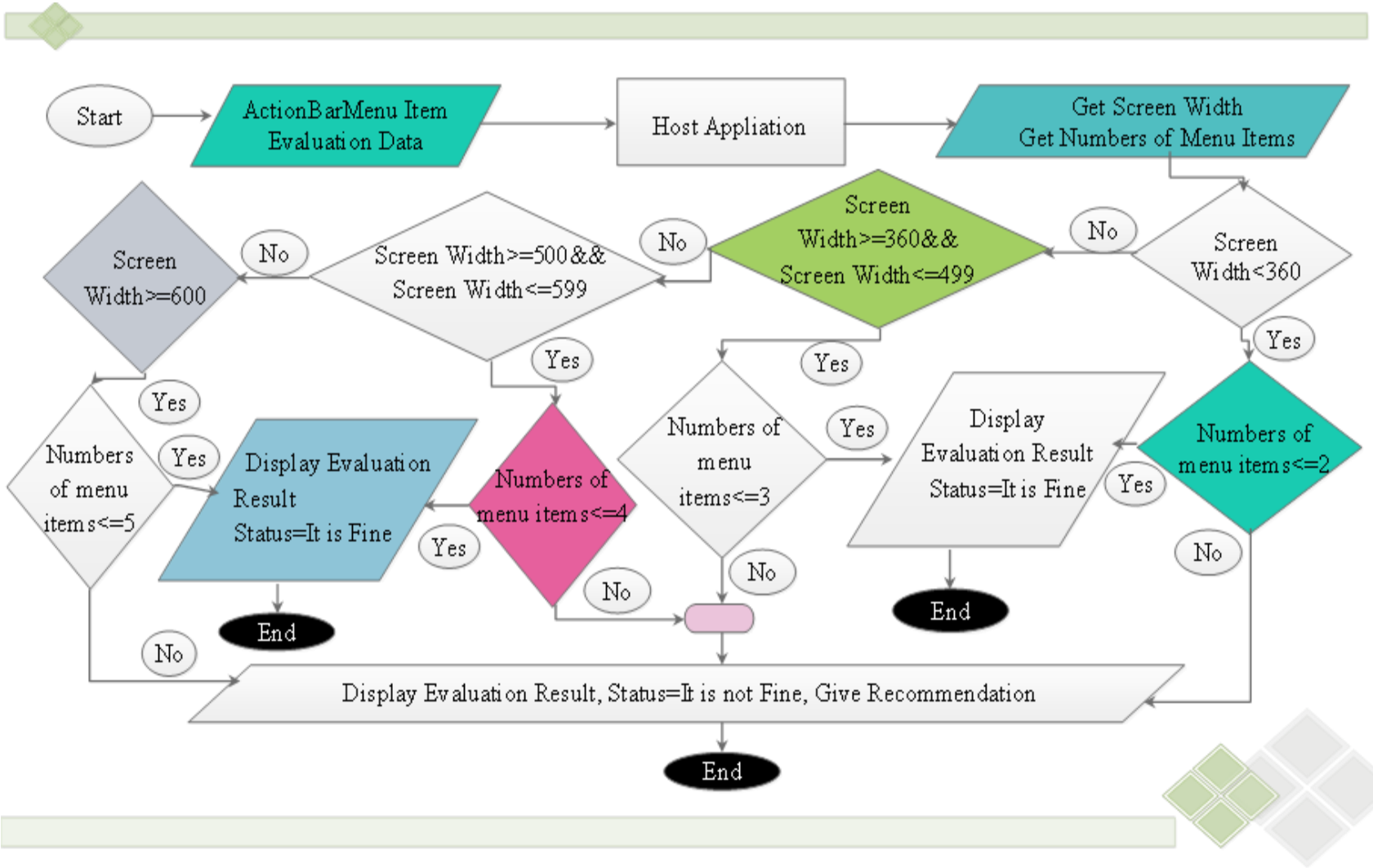


Figure 5:7Action Bar Menu Item Evaluation Flow Charts

5.4.3. Other User Interface Element Evaluation Frameworks

The other user interface evaluation frameworks are part of the proposed framework. But, these are not implemented in this prototype. However, we described the design of these user interface element evaluation components. The proposed design framework is included basic user interface elements which are descried in *chapter four*.

5.4.3.1. Color and Contrast Evaluation framework

The proposed color and contrast evaluation framework designed to evaluate color and contras of user interface based on the guidelines which defined in *chapter four*. The color and contrast information like back ground color, and text color send into the color evaluation framework. The proposed framework evaluated background and text color ratio based on the guidelines and generate evaluation result and recommendation solution when detected color and contrast problems.

5.4.3.2. Menu Design Evaluation Framework

The menu design evaluation framework consists of basic components in order to evaluate the menu design such as screen density for evaluate the menu item size. The menu design evaluation components are inserted in the host application. Then, the framework can evaluate menu item height based on the mobile screen density. Additionally, the proposed framework evaluates the alignment of menu item from screen layout based on guidelines.

5.4.3.3. Tab Design Evaluation Framework

Android tab design evaluation framework can perform based on the following ways. Tab evaluation component source codes are inserted into host application. Basically, the tab evaluation component consists of screen size, tab size, alignment, color and text properties based on tab design guidelines [15] [45]. The framework should check the type of tab (i.e scalable or fixed). When the tab is fixed tab then calculate tab width otherwise evaluate tab design based on Android guidelines which are described in *chapter four*. Finally, the proposed framework generates evaluation result with recommendation solution when detected tab design problems.

5.4.3.4. Button Design Evaluation Framework

The proposed user interface elements usability evaluation framework has provided button design properties like button size, alignment, and color based on type of button. When the host application has an overflow action button the framework should evaluate the button size based on mobile screen width.

5.4.3.5. Browser Widget Design Evaluation Framework

The automated user interface evaluation framework provided browser widget evaluation based on Android guidelines. The browser widget evaluation framework performed such calculate numbers of browser widget available in the host application then display evaluation result. When numbers of browser widget greater than two, the proposed framework displayed recommendation solution likes using pagination.

5.4.3.6. Map Widget Design Evaluation Framework

The map widget evaluation process is similar to browser widget. But, the maximum number of browser widget should be one on the user interface. The proposed map evaluation framework can display evaluation result with recommendation solution when the number of map widget greater than one.

5.4.3.7. Segmented UI Evaluation Framework

The proposed framework design for evaluate segmented user interface components. First, the framework verified the segmented user interface is either complex or simple segment based on numbers of widgets inside the segmented user interface. Then, it is evaluated each individual user interface widget based on their number of rows (records). When segmented user interface widget has consisted more than number of rows or records specified in the guidelines, then the framework will be reported and give recommendation solution like the element should be pagination into multiple user interface widgets.

5.4.3.8. Orientation Support Evaluation Framework

The proposed orientation evaluation framework designed to evaluate user interface element margin and padding size when mobile screen orientation changed. This evaluation framework should calculate the proportional of the screen height and width. The framework should check

margin and padding value change based on the proportional value or not. The proposed framework also provided recommendation to the developer to use "On Orientation" skins used for both orientations.

5.4.3.9. Mobile Icon Design Evaluation Framework

The proposed icon design evaluation framework used to mobile device information like screen width, height, density and Android OS based on type of icons. The proposed icon evaluation components are used to collect and send icon information from host application into their proposed evaluation framework.

- ***Menu Icons Evaluation Framework***: Menu icon evaluation framework consists of different types of icon evaluation elements including menu icon dimension, style, and effect, shadow and color plate information. The proposed framework evaluated the menu icon based their screen density and Android version.
- ***Action Bar Icons Evaluation Framework***: An action bar icons evaluation framework is designed to evaluate the dimension of action bar icon based on mobile screen densities. The actions bar icon dimension send into the proposed framework. Then, the proposed framework evaluated the action bar icon based on the screen density and provided recommendation solution.
- ***Tab Icons Evaluation Component***: This tab icon evaluation framework is designed for evaluating the tab icon based on the guidelines which descried in chapter four. Tab icon evaluation attributes like icon dimension, color, style, effects are send into the proposed evaluation framework. The proposed evaluation framework evaluated tab icon dimension based on mobile screen density and tab icon color, style, effects based on Android OS version.

CHAPTER SIX

6. Implementation of the Proposed Framework

This chapter described the prototype implementation of the proposed framework. Each user interface usability evaluation elements are implemented in a separate way. Generally, the proposed usability evaluation framework has three major implementation modules such as user interface evaluation framework home screen, user interface evaluation component and user interface usability evaluation framework. Home screen is the first user interface of the proposed framework which used to call host application or mobile device evaluation framework. User interface evaluation components are the proposed user interface evaluation attributes and methods which are added into the host application. The user interface usability evaluation framework has six proposed framework library activity classes in *mylibrary* package. This includes*MobileDeviceInformation, headerandFooterEvaluation, LineLengthEvaluation, ButtonTypographyEvaluation, MarginEvaluation,* and *ActionBarMenuItemEvaluation* Android library activity classes.

6.1. Common Implementation Activities

The proposed user interface evaluation framework has common activities applied in each user interface element evaluation. These common activities are described in the following ways by using header and footer host application and its evaluation framework.

- Host application information passed into user interface evaluation framework by intent object. This performed by *onClick* event handler method which is defined inside their xml file.

```
Bundle bundle=new Bundle();
public void result(View v)
{
    onResume();
    Intent intent=new Intent(this,HeaderAndFooterEvaluation.class);
    Bundle bundle=new Bundle();
    bundle.putInt("footerHeight",footerHeader);
    bundle.putInt("headerHeaght",headerHeight);
    intent.putExtras(bundle);
    startActivity(intent);
}
```

- Accessed user interface evaluation element information's' which are sent from host application

```
Bundle bundle = getIntent().getExtras();
int headerHeight,footerHeight;
//Retrieve header and footer height from evaluated uer interface
headerHeight=bundle.getInt("headerHeaght");
footerHeight=bundle.getInt("footerHeight");
```

- Display evaluation result on Android emulator, mobile device and computer device.

```
currentValueText.setText ("Current Height  "+ currentHeaderAndFooterHeight +"\t px");
expectedValueText.setText ("Expected Height=" +expectedHeaderAndFooterHeight +"\t px");
Log.i("Expected"," Header and Footer Height= "+expectedHeaderAndFooterHeight +" px");
Log.i("Current"," Header and Footer Height=" + currentHeaderAndFooterHeight +" px");
```

- When move from one application activity to the other application activity, the application activity package should be called by

getPackageManager().getLaunchIntentForPackage("packageName") intent method.

```
public void headerAndFooterEvaluation(View view)
{
 Intent headerAndFooterEvaluationIntent=getPackageManager().getLaunchIntentForPackage
        ("com.javacodegeeks.android.androidscrollablecontent");
   startActivity(headerAndFooterEvaluationIntent);
}
```

- The automated user interface usability evaluation library class should be import in the host application. `import com.example.mylibrary.HeaderAndFooterEvaluation;`
- User interface usability evaluation library class should be called inside the host application.

```
Intent intent=new Intent(this,HeaderAndFooterEvaluation.class);
startActivity(intent);
```

- The proposed framework activity class must be added into host application manifest xml file. `<activity android:name="com.example.mylibrary.HeaderAndFooterEvaluation"/>`
- The proposed framework should be added dependency into host application. This is the process of proposed evaluation framework integrated with host application at compile time. This activity required some subsequent procedures such as File→Project Structure→select user interface module→ dependency→select the framework in the scope of compile→ok.

6.2. The Proposed Framework Implementation

The proposed prototype has three major implementation modules such as home screen, user interface evaluation component and user interface evaluation framework.

6.2.1. User Interface Usability Evaluation Home Screen

The proposed framework has one major user interface which is called home screen. Home screen has six buttons. These buttons are used to call host application and mobile device evaluation framework. Additionally, home screen user interface has help menu items. Help menu items are used to describe basic information about each evaluated user interface evaluation components. The home screen is illustrated in below diagram and it descried in the next section.

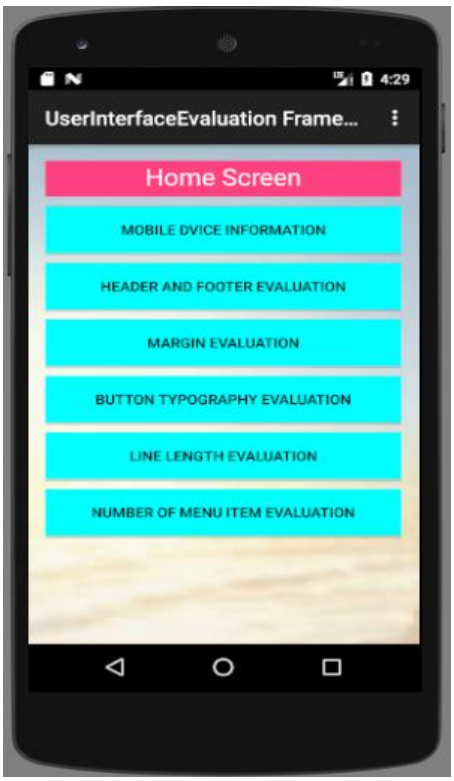


Figure 6:1 Proposed User Interface Usability Evaluation Framework Home Screen

- ***Mobile Device Information:*** is a home screen button which used to call mobile device information library class.
- ***Header and Footer Evaluation***: is a home screen button that used to pass header and footer evaluated host application.
- ***Line Length Evaluation***: is a home screen button which used to call line length evaluated host application.
- ***Button Typography Evaluation***: is a home screen button that used to call button typography evaluation host application.

- ***Margin Evaluation***: is home screen button that used to call margin evaluation host application user interface.
- ***Number of Menu Item Evaluation***: is a home screen button which used to call evaluated action bar menu item host application.

6.2.2. User Interface Evaluation Components Implementation

User interface evaluation components are part of the proposed framework. The evaluation source codes are inserted in the host application. Some source code may modify based on evaluated host application. User interface evaluation components are used to collect user interface information and pass collected information into user interface evaluated framework. Generally, we descried five user interface evaluation components implementation with sample source code in bellow sections.

6.2.2.1. Header and Footer Evaluation Component Implementation

This component has header and footer height information. When header and footer evaluation, the developer should be added below source code into the host application.

- The evaluated user interface header and footer height are gotten by *onResume()* method.

```
@Override
protected void onResume() {
    super.onResume();
    RelativeLayout headerView = (RelativeLayout) findViewById(R.id.header);
    RelativeLayout footerView =(RelativeLayout)findViewById(R.id.footer);
    headerHeight = headerView.getHeight();
    footerHeader= footerView.getHeight();
}
```

- Send header and footer height into *headerAndFooterEvaluation* library activity class by intent object.

Generally, the layout and its resource name may be modified by the developer during adding into host application. But, the other header and footer evaluation source code doesn't modify because it affected the evaluation framework.

6.2.2.2. Margin Evaluation Component Implementation

Mobile application has different type of components on their user interface like widget, view and etc. Mobile application has various numbers of user interface elements from user interface to the

other user interface. Hence, the prototype margin evaluation component has consists of four user interface elements' information such as name, margin values, sizes and user interface layout size. We described for one evaluated user interface element implementation process in below and the three evaluated elements have similar procedures.

- Specified user interface evaluation elements.

```
Button userInterfaceElementOne;
userInterfaceElementOne = (Button) findViewById(R.id.button1);
```

- Accessed user interface element position values from host application user interface layouts.

```
elementOneLeft = userInterfaceElementOne.getLeft();
elementOneTop= userInterfaceElementOne.getTop();
elementOneRight = userInterfaceElementOne.getRight();
elementOneBottom= userInterfaceElementOne.getBottom();
```

- Accessed a layout size which is hold evaluated user interface elements.

```
final RelativeLayout containerLayout = (RelativeLayout) findViewById
        (R.id.activity_margin_example);
int  layoutHeight = containerLayout.getMeasuredHeight();
int  layoutWidth = containerLayout.getMeasuredWidth();
```

- Accessed user interface evaluated element height and width values.

```
//Get  user interface element one size such as Width and Height
int elementOneWidth= userInterfaceElementOne.getWidth();
int elementOneHeight= userInterfaceElementOne.getHeight();
```

- Accessed user interface evaluated elements margin values.

```
// Get user interface element One Margin values
View viewElementOne = findViewById(R.id.button1);
ViewGroup.MarginLayoutParams marginLayoutParamsViewOne = (ViewGroup.MarginLayoutParams)
        viewElementOne.getLayoutParams();
int bottomMarginElementOne = marginLayoutParamsViewOne.bottomMargin;
int rightMarginElementOne = marginLayoutParamsViewOne.rightMargin;
int topMarginElementOne = marginLayoutParamsViewOne.topMargin;
int leftMarginElementOne = marginLayoutParamsViewOne.leftMargin;
```

- Accessed user interface evaluated element name.

```
elementOneName = userInterfaceElementOne.getText().toString();
elementTwoName = userInterfaceElementTwo.getText().toString();
elementThreeName = userInterfaceElementThree.getText().toString();
elementFourName = userInterfaceElementFour.getText().toString();
```

The above user interfaces element information should pass into *MarginEvaluation* framework. This source code should add on the host application during evaluation session. However, layout type and their resource identification name may be modified based on user interface element. However, the intent string name and variable name doesn't modify because it affects evaluation framework results.

6.2.2.3. Button Typography Evaluation Component Implementation

The proposed button evaluation component implementation has language code, language script type, language name, and text style and button name information. The language evaluation source code should add into the evaluated host application.

```
public void languageEvaluation(View view)
{  userInterfaceElement =(Button) findViewById(R.id.languageResult);
    int elementTextSize = (int) userInterfaceElement.getTextSize();
   Typeface buttontypface= userInterfaceElement.getTypeface();
    CharSequence buttonName= userInterfaceElement.getText();
    Bundle bundle=new Bundle();
    bundle.putString("languageType",languageToLoad);
    bundle.putInt("TextSize", elementTextSize);
    bundle.putInt("style",buttontypface.getStyle());
    bundle.putCharSequence("buttonName",buttonName);
    Intent intent=new Intent(this,LanguageEvaluation.class);
    intent.putExtras(bundle);
    startActivity(intent);
}
```

- Finally, the host application information is sent into *ButtonTypograpghyEvaluation* evaluation library activity class.

6.2.2.4. Line Length Evaluation Component Implementation

The line length evaluation component consists of line information which used to collect host application text information like total numbers of characters, total numbers of lines, start and end line and numbers of characters per line. The proposed line length evaluation component implemented based on supported both text view and edit text line length. Generally, the following source code added into the evaluated host application during evaluation session.

- Gotten total numbers of line, and total numbers of characters.

```
int numberOfLine=textView2.getLineCount();
int totalNumberOfCharacter=(textView2.length()-numberOfLine)+1;
```

- Gotten start line, end line and numbers of character per line for each line.

```
for(int i=0;i<numberOfLine;i++) {
    startLineValue[i] = textView2.getLayout().getLineStart(i);
    if (((i + 1) == numberOfLine)) {
        endLineValue[i] = textView2.getLayout().getLineEnd(i);
        numbersOfCharacterPerLine[i] = endLineValue[i] - startLineValue[i];
    } else {
        endLineValue[i] = textView2.getLayout().getLineEnd(i)-1;
        numbersOfCharacterPerLine[i] = endLineValue[i]-startLineValue[i];
    }
}
```

- The above line information's are sent into *LineLengthEvaluation* framework.

6.2.2.5. Acton Bar Menu Item Evaluation Component Implementation

The proposed action bar menu item evaluation component consists of menu When evaluated numbers of menu item on the action bar should added some source code into the evaluated host application.

- Accessed numbers of menu item valuable on the action bar.

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main, menu);
    numbersOfMenuItems=menu.size();
    return true;
}
```

- Send action bar item evaluation information into *ActionBarMeniItemEvaluation* framework.

6.2.3. User Interface Usability Evaluation Framework Implementation

User interface usability evaluation framework has six library classes with its user interface. This proposed framework module name is called *mylibrary*. The proposed user interface evaluation framework has a package that is called *com.example.mylibrary*. Each user interface evaluation framework class is described in section below.

6.2.3.1. Mobile Device Information Library

The mobile device information library is a part of proposed framework which used to display mobile device information. It consists of mobile display information attributes, and four

methods. Basically, the mobile device information can be display when declared *DisplayMetics* class. *DisplayMetrics is* a public class extends from an object. This used to access mobile device display information such as its size, density, and OS. The *DisplayMetrics* should initialize an object that is called *getWindowManager().getDefaultDisplay().getMetrics(DisplayMetrics)*. The mobile device evaluation library implementation processes are described below.

- Accessed mobile screen information such as scaling factor, width, height, and screen density respectively.

```
DisplayMetrics metrics=new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(metrics);

screenDensity =metrics.density;
screenWidth=metrics.widthPixels;
screenHight=metrics.heightPixels;
screenDpi=metrics.densityDpi;
```

- Accessed mobile OS information including mobile manufacturer, mobile model, name version and version release.

```
mobileManifacturer= Build.MANUFACTURER;
mobileModel=Build.MODEL;
OSVersionRelease=Build.VERSION.RELEASE;
OSversion=Build.VERSION.SDK_INT;
OSVersionName=Build.VERSION_CODES.class.getFields()
        [android.os.Build.VERSION.SDK_INT].getName();
```

- Calculated physical screen size based on *Equation 2:1 Calculate Screen Size* of mobile device in inch and the result can display in two digit decimal formats.

```
public double getScreenSizeInInch()
{
    mobileWidthInInch = Math.pow(screenWidth / screenDpi, 2);
    mobileHeightInInch = Math.pow(screenHight/ screenDpi, 2);
    screenSizeInch = Math.sqrt(mobileWidthInInch + mobileHeightInInch);
    //to convert decimal number into two digit after point
    DecimalFormat twoDForm = new DecimalFormat("#.##");
    resultInDecimalFormat =Double.valueOf(twoDForm.format(screenSizeInch));
    return resultInDecimalFormat;
}
```

- Identified screen density type (class) of mobile device based on their screen dpi.

```
public String getScreenDensityType() {
    if (screenDpi > 0 && screenDpi <= 120) {
        screendensityType = "ldpi";
    }
    else if (screenDpi > 120 && screenDpi <= 160) {
        screendensityType = "mdpi";
    }

    return screendensityType;
}
```

- Identified screen size class of a mobile device based on their mobile screen size.

```
public String getScreenSizeType(){
    if (getScreenSizeInInch()>=2 && getScreenSizeInInch()<=3){
        screenSizeType="Small Screen";
    }
    else if (getScreenSizeInInch()>=3&& getScreenSizeInInch()<=5)
    {
        screenSizeType="Normal Screen";
    }


    return screenSizeType;
}
```

- Display mobile device information by mobile phone and computer devices.

6.2.3.2. Header and Footer Evaluation Library

The header and footer evaluation library consists of screen height, header and footer height values. This library can calculate current, expected and over height of header and footer of host application user interface based on mobile screen height.

```
int  expectedHeaderAndFooterHeight;
expectedHeaderAndFooterHeight= (int) Math.round((screenHeight*0.3));
int currentHeaderAndFooterHeight =headerHeight+footerHeight;
int  overHeaderAndFooterHeight;
overHeaderAndFooterHeight = Math.round(
        (currentHeaderAndFooterHeight - expectedHeaderAndFooterHeight));
```

- Checked the status of host application header and footer value either conformance the guidelines or not based on the expected and current height value. When the status is fine, then the result display in green color. Whereas the result isn't fine the status can display in red color with minimize height value from header and\or footer.

```
String status;
if(currentHeaderAndFooterHeight <= expectedHeaderAndFooterHeight) {
    statusText.setTextColor(Color.parseColor("#00FF00"));
    status = "It is Fine";
    statusText.setText("Status="+status);
}
else{
    statusText.setTextColor(Color.parseColor("#FF0000"));
    status=" Over Height must be minimize by=";
    statusText.setText("Status="+status+ overHeaderAndFooterHeight +"px");
}
```

- Display evaluation results on display devices.

6.2.3.3. Margin Evaluation Library

The margin evaluation library consists of user interface information and evaluated element attributes. Such as user interface layout size and each evaluated element name, sizes, margin values, position values. Basically, the space between element and margin values based on evaluated element alignment on user interface and free layout space.

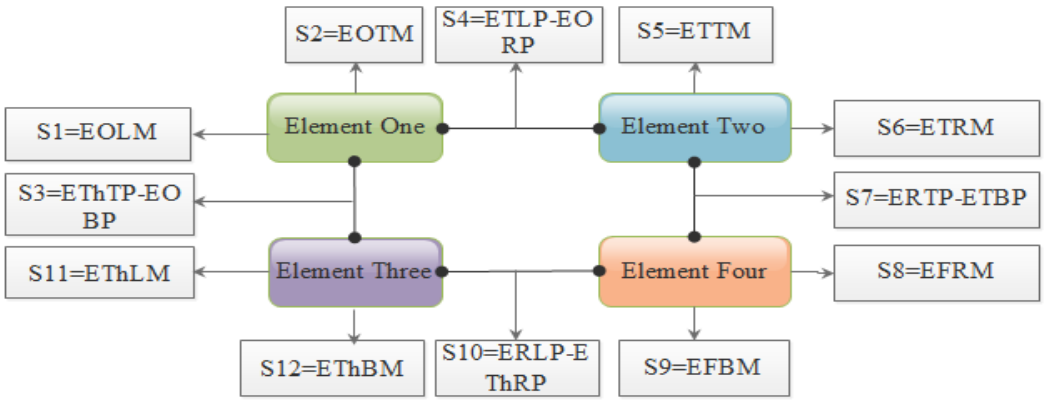


Figure 6:2 Calculate Space between User Interface Elements

S1	***S2***	***S3***	***S4***	***S5***	***S6***	***S7***	***S8***	***S9***	***S10***	***S11***	***S12***	***S13***
Space1	Space1	Space1	Space1	Space1	Space1	Space1	Space1	Space1	Space1	Space1	Space1	Space1

The above Figure 6:2and in *Figure 9: 1* appendix (B) diagrams illustrated that sample user interface element alignment and it calculates a space between user elements. The above abbreviations are expanded in table below.

Abb.	*Expand Form*	*Abb.*	*Expand Form*
EOLM	Element One Left Margin	EOLP	Element One Left Position
EOTM	Element One Top Margin	EOTP	Element One Top Position
EORM	Element One Right Margin	EORP	Element One Right Position
EOBM	Element One Bottom Margin	EOBP	Element One Bottom Position
ETLM	Element Two Left Margin	ETLP	Element Two Left Position
ETTM	Element Two Top Margin	ETTP	Element Two Top Position
ETRM	Element Two Right Margin	ETRP	Element Two Right Position
ETBM	Element Two Bottom Margin	ETBP	Element Two Bottom Position
EThLM	Element Three Left Margin	EThLP	Element Three Left Position
EThTM	Element Three Top Margin	EThTP	Element Three Top Position
EThRM	Element Three Right Margin	EThRP	Element Three Right Position
EThBM	Element Three Bottom Margin	EThBP	Element Three Bottom Position
EFLM	Element Four Left Margin	EFLP	Element Four Left Position
EFTM	Element Four Top Margin	EFTP	Element Four Top Position
EFRM	Element Four Right Margin	EFRP	Element Four Right Position
EFBM	Element Four Bottom Margin	EFBP	Element Four Bottom Position

The margin evaluation library can calculate total element size, free layout space, expected space between elements and expected margin values based on user interface element alignment.

- When all evaluated user interface elements are aligned vertically.

```
totalWidgetHeight = widgetOneHeight + widgetTwoHeight + widgetThreeHeight + widgetFourHeight;
totalWidgetWidth = (widgetOneWidth + widgetTwoWidth + widgetThreeWidth + widgetFourWidth) / 4;
freeLayoutSpaceWidth = layoutWidth - totalWidgetWidth;
freeLayoutSpaceHeight = layoutHeight - totalWidgetHeight;
leftAndRight = freeLayoutSpaceWidth * .02;
topAndright = freeLayoutSpaceHeight * .01;
```

- When all evaluated user interface elements are aligned horizontally.

```
totalWidgetHeight = (widgetOneHeight + widgetTwoHeight + widgetThreeHeight + widgetFourHeight) / 4;
totalWidgetWidth = widgetOneWidth + widgetTwoWidth + widgetThreeWidth + widgetFourWidth;
freeLayoutSpaceWidth = layoutWidth - totalWidgetWidth;
freeLayoutSpaceHeight = layoutHeight - totalWidgetHeight;
leftAndRight = freeLayoutSpaceWidth * .02;
topAndright = freeLayoutSpaceHeight * .01;
```

- When evaluated elements are arranged in two vertically and two horizontally.

```
totalWidgetHeight = (widgetOneHeight + widgetTwoHeight + widgetThreeHeight + widgetFourHeight) / 2;
totalWidgetWidth = (widgetOneWidth + widgetTwoWidth + widgetThreeWidth + widgetFourWidth) / 2;
freeLayoutSpaceWidth = layoutWidth - totalWidgetWidth;
freeLayoutSpaceHeight = layoutHeight - totalWidgetHeight;
leftAndRight = freeLayoutSpaceWidth * .02;
topAndright = freeLayoutSpaceHeight * .01;
```

- When evaluated elements are aligned vertically and horizontally in different ways. Example:

```
totalWidgetHeight = (widgetOneHeight+widgetTwoHeight + widgetThreeHeight + widgetFourHeight)/2;
totalWidgetWidth = (widgetOneWidth + widgetTwoWidth+widgetThreeWidth+widgetFourWidth)/3;
freeLayoutSpaceWidth = layoutWidth - totalWidgetWidth;
freeLayoutSpaceHeight = layoutHeight - totalWidgetHeight;
leftAndRight =(int) Math.round(freeLayoutSpaceWidth * .02);
topAndBottom = (int)Math.round(freeLayoutSpaceHeight * .01);
```

- Display evaluation result like each space value, free layout space size, and etc.

6.2.3.4. Button Typography Evaluation Library

The button typography evaluation library has button evaluation attributes. This includes language script type, language type, text size, text style, font type, and button name. Generally, we defined basic implementation process and sample source code in the following ways.

- Checked a button name is capital or small letter by regular expression pattern matcher.

```
String patternString = "[A-Z]+";
Pattern pattern = Pattern.compile(patternString, Pattern.UNICODE_CASE);
Matcher matcher = pattern.matcher(elementName);
boolean isMatched = matcher.matches();
```

- Android studio user interface element information's are displayed in pixel form. Hence, the current button size should be converted into density-independent pixel (dp) form by *Equation 2:4*

```
DisplayMetrics metrics = Resources.getSystem().getDisplayMetrics();
int CurrentTextSizeInDp = currentbuttonTextSizeInPixel / (metrics.densityDpi / 160);
```

- Determine the button status of expected button text size, font type and text style based on language script type.

```
statusText=(TextView)findViewById(R.id.statusTextView);
if (LanguageScript.equals("English-like")) {
    expectedButtonTextSize = 14;
    if ((CurrentTextSizeInDp == expectedButtonTextSize) || isMatched == true) {
        if (CurrentTextSizeInDp == expectedButtonTextSize && isMatched == true) {
            statusText.setTextColor(Color.parseColor("#00FF00"));
            status = " Button text size and font type are fine";
        } else if (CurrentTextSizeInDp == expectedButtonTextSize && isMatched != true) {
            statusText.setTextColor(Color.parseColor("#FF0000"));
            status = "Button text  size fine but font type  isn't capital";
        } else {
            statusText.setTextColor(Color.parseColor("#FF0000"));
            status = "Button text  size doesn't fine but font type is capital";
        }
    } else {
        statusText.setTextColor(Color.parseColor("#FF0000"));
        status = " Button text size and font type aren't fine";
    }
```

6.2.3.5. Line Length Evaluation Library

Basically, this library implementation has line attribute values such as total numbers of lines, number of character per lines, start and end lines. These line evaluation attribute values are received host application line information by array form.

```
int startLineValue[] = bundle.getIntArray("startLine");
int endLineValue[] = bundle.getIntArray("endLine");
int numberOfLine = bundle.getInt("numberOfLine");
int totalNoOfCharacter=bundle.getInt("totalCharacter");
int numberOfCharacter[] = bundle.getIntArray("numbersOfCharacter");
```

- Identified user interface evaluation element text line type based on number of line in the paragraph.

```
if(numberOfLine<=2)
{
    typesOfLine="Short Line";
}
else
{
    typesOfLine="Body Text";
}
```

- Evaluate the line length and identify the status of each text paragraph. Specifically, line length classified into three major categories such as too narrow, ideal range and too wide. When the evaluation result has ideal range the line length is fine, otherwise the evaluated line isn't fine state. So the developer must be minimize or maximize the character for evaluated line.

```
for (int i=0;i<numberOfLine;i++) {
    // Log.i("Line",(i+1)+"");
    if(numberOfLine<=2)
    {
        if(numberOfLine==1) {
            if (numberOfCharacter[i] > 30) {
                status="Two Wide";
            } else {
                status="Ideal Range";
            }
        }
        else {
            if ((numberOfCharacter[i] >= 1) && (numberOfCharacter[i] <= 15)) {
                status="Too Narrow";
            } else if (numberOfCharacter[i] > 15 && numberOfCharacter[i] <= 30) {
               status="Two Wide";
           }
       }
```

- Display evaluation result such as total number of lines, number of characters per line, start and end line, too narrow, too wide, and unnecessary lines.

```
Log.i("Unnecessary line ","Must be Removed");
for(int i=0;i<numberOfLine;i++) {
    if (i >= 1 && i < numberOfLine - 1) {
        if ((numberOfCharacter[i - 1] == 0 && numberOfCharacter[i] == 0)) {
            Log.i(" Unnecessary Lines", (i + 1) + "\t \t" + "Start line=" + startLineValue[i] +
                    "\t \t" + "End line=" + endLineValue[i] + "\t \t" + "No Of Character " +
                    numberOfCharacter[i]);

        }
    }
}
```

6.2.3.6. Action Bar Menu Item Evaluation Library

The action bar evaluation library consist action bar evaluation attributes such as number of menu items, and screen width in dp values. The numbers of menu items value passed from the host application. We described evaluation process with sample source code in below.

- The screen width should be converted from px into dp.

```
float result = (float) (screenDpi / 160.0);
int  screenwidthInDp = Math.round(screenWidth / result);
```

- Assigned the expected menu item based on mobile screen width.

```
if (screenwidthInDp <= 360) {
    actualNumberOfIcons = 2;
} else if (screenwidthInDp >= 360 && screenwidthInDp <= 499) {
    actualNumberOfIcons = 3;
} else if (screenwidthInDp >= 500 && screenwidthInDp <= 599)
    actualNumberOfIcons = 4;
else if (screenwidthInDp >= 600) {
    actualNumberOfIcons = 5;
}
```

- Calculated numbers of over menu items.

```
numberOfOverMenuItems = numbersOfMenuItems - expectedNumberOfMenuItems;
```

- Evaluated the menu bar status. When over menu item is negative or equal to zero, then action bar menu items are fine.

```
if (numbersOfMenuItems>expectedNumberOfMenuItems)
{ status="numbers of menu items are minimized by ";
    statusText.setTextColor(Color.parseColor("#FF0000"));
    statusText.setText("Status="+status+numberOfOverMenuItems);
}
else
{
    status="Current menu items are equal to expected menu items";
    statusText.setTextColor(Color.parseColor("#FF0000"));
    statusText.setText("Status="+status);
}
```

- Display evaluation result such as expected numbers of menu item, current menu items, number over menu item.

CHAPTER SEVEN

7. Evaluation, Result and Discussion

In this chapter, we described the evaluation of the proposed automated user interface usability evaluation prototype on mobile devices and various user interfaces. The proposed framework evaluation process has three consecutive activities: such as preparation, integration and generate evaluation results. The preparation is the first activity which includes the downloading of the Android projects to prepare the sample user interface applications. The integration is the second process which performed different activities. It includes the insertion of the evaluation source code into the sample user interface and integrates the proposed framework with the host application. Generate evaluation result is the last activity that runs the host application on Android studio emulators or mobile devices. Finally, the proposed framework displays the evaluation result both on mobile device and emulators. We evaluated the proposed prototype should be evaluated the host application based on the guidelines or not. We are evaluated each user interface element in different host application and in different mobile screen size. We discussed the evaluation processes and the results as follows.

7.1. Mobile Device Information Evaluation Result

Mobile device evaluation framework can display basic mobile device information like screen and OS information. The evaluation was performed on various types of Android mobile devices and emulators. However, we illustrated for only seven different mobile devices evaluation results in next table and two sample evaluation diagrams. This evaluation result shown that the mobile information's are similar results. For example, the first emulator screen size is 4.33 inch then the screen type is normal. Whereas, diagram shown that the screen size 5.25 inch. This screen size is greater than five so screen size type also large.

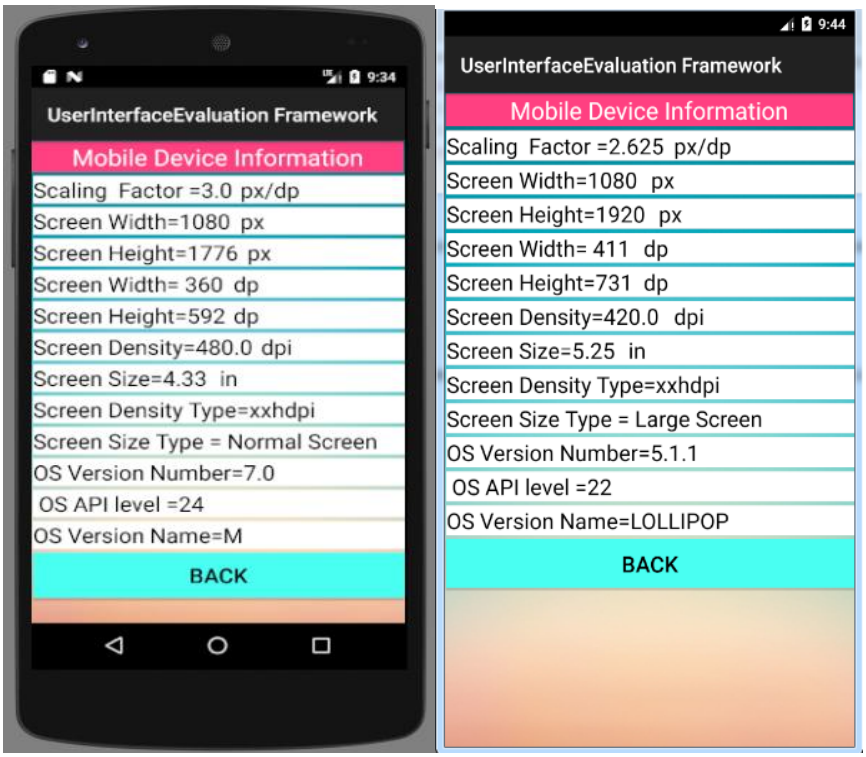


Figure 7:1 Android Mobile Device Evaluation Results

In below evaluation results are compared with the theoretical values of mobile device which described in *Table 2-1* , *Table 2-2*, and *Table 2-3* . Additionally, some mobile device information's are evaluated based on their calculation results. We are used two formulas such as *Equation 2:1* for screen size and *Equation 2:4* for screen width and height in dp. The evaluated and expected smart phone information described in below tables respectively. According to below evaluation and expected mobile information results. We concluded that most of the evaluated mobile device information's are similar to the expected mobile information. However, some information's have different result from the expected value. This difference came from the approximation methods of the proposed framework. Hence, it is approximated the calculated element result by using math's rounded (approximation) method such as screen width and screen height in dp. However, this variation didn't effect on mobile usability evaluation. Additionally, Android OS version 7.0 mobile device version name is "Nougat" according to *Table 2-1*. But, the proposed framework display M .M is an abbreviation name of Marshmallow.

Table 7-1 Smart Phone Device OS Evaluation Information Results

	Scaling factor(px/dp)	***Width (px)***	***Height (px)***	***Width (dp)***	***Height (dp)***	***Density (dpi)***	***Size (in)***	***Density type***	***Size type***	***Version number***	***API level***	***Version name (code)***
1	3.0	1080	1776	360	592	480.0	4.33	xxhdpi	Normal	7.0	24	M
2	2.625	1080	1920	411	732	420.0	5.25	xxhdpi	Large	5.1.1	22	LOLLIPOP
3	1.5	480	854	320	569	240.0	4.98	hdpi	Normal	4.4.4	19	KITKAT
4	2.0	2560	1504	1280	752	320.0	9.28	xhdpi	Tablet	7.0	24	M
5	2.0	720	1280	360	640	320.0	4.59	xhdpi	Normal	6.0	23	LOLLIPOP-MR1
6	3.0	1080	1806	360	602	480.0	4.38	xxhdpi	Normal	7.0	24	M
7	1.5	480	800	320	533	240.0	3.89	hdpi	Normal	4.2.2	17	JELLYBEAN

Table 7-2 Smart Phone Device expected Result

	Width(dp)	***Height(dp)***	***Size(in)***	***Density type***	***Size type***	***API level***	***Version name (code)***
1	360	592	4.33	xxhdpi	Normal	24-25	Nougat
2	411.43	731.43	5.25	xxhdpi	Large	21-22	Lollipop
3	320	569	4.98	hdpi	Normal	19	Kit Kat
4	1280	752	9.28	xhdpi	Tablet	24-25	Nougat
5	360	640	4.59	xhdpi	Normal	23	Marshmallow
6	360	602	4.38	xxhdpi	Normal	24-25	Nougat
7	320	533.33	3.89	hdpi	Normal	16-18	Jellybean

7.2. Header and Footer Evaluation Results

This proposed framework evaluated in different host applications and various types of mobile devices. The user interface header and footer evaluation results are shown in *Figure 9: 2* (appendix C) and in figure below.

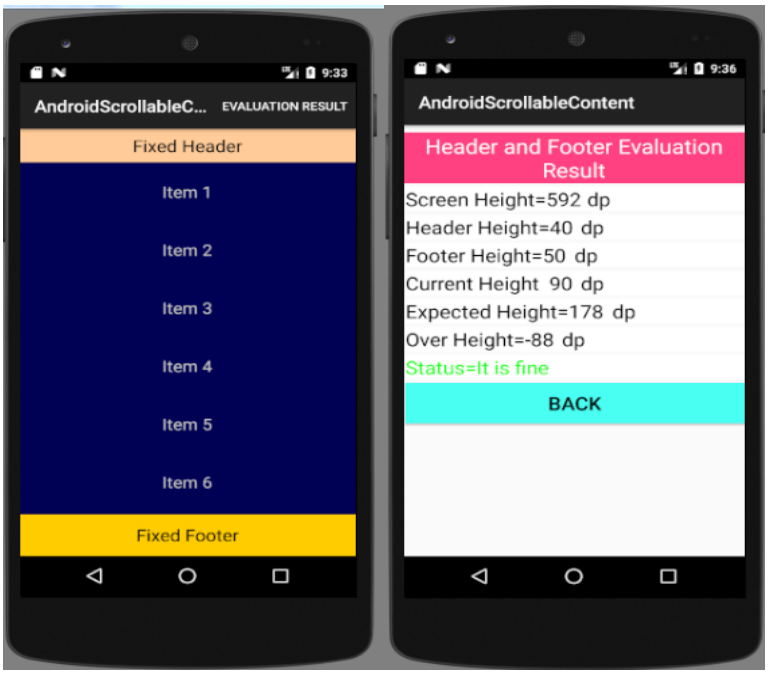


Figure 7:2 Heard and Footer Evaluation Host Application and Result

This below table is described header and footer evaluation framework evaluation test case.

Table 7-3 Header and Footer Evaluation Test Case

	Test Case 1	***Test Case 2***
Current height<= Expected height	Yes	No
Over height	Negative	Positive
Status	It is fine	It is not fine.
Solution	There is no change.	Header and/or footer height should be minimized by over height value.

Generally, the header and footer evaluation performed on three user interfaces and different mobile devices. The evaluation results are illustrated in table below.

Table 7-4 Header and Footer Evaluation Result

	Screen height (dp)	***Header height (dp)***	***Footer height (dp)***	***Current Height (dp)***	***Expected Height (dp)***	***Over Height (dp)***	***Status***
1	592	40	50	90	178	-88	It is fine
2	592	80	90	170	178	-8	It is fine
3	592	90	90	180	178	2	It is not fine and It should be minimize by 2 dp.
4	533	80	90	170	160	10	It is not fine and It should be minimize by 10 dp.
5	533	40	50	90	160	-70	It is not fine
6	533	90	90	180	160	20	It is not fine and It should be minimize by 10 dp.
7	569	80	90	170	171	-1	It is fine
8	569	40	50	90	171	-81	It is fine
9	569	90	90	180	171	9	It is not fine and It should be minimize by 10 dp.

We calculated the current and expected height value by *Equation 5:1* formula *which* is described in *chapter five*. Generally, the above header and footer evaluation results are similar to theoretical header and footer evaluation based on the above test case.

7.3. Button Typography Evaluation Result

The proposed button typography evaluation framework used to evaluate button typography information. Basically, the proposed framework evaluated user interface button information based on language script type. The proposed framework provided summarized evaluation results such as language code, language script type, language name, and button text size and text style. We proposed typography button evaluation test cases based on *Table 4-1* defined in chapter four. Basically, the button typography evaluation has three test cases which are defined in the following tables.

Table 7-5 Multi-language Button Evaluation Test Case

	Test Cases			
	Case 1	***Case 2***	***Case 3***	***Case 4***
Language script type	English-like	English-like	English-like	English-like
Curent text size	14	!=14	14	!=14
Button name	All caps	All caps	Not Caps	Not Caps
Expected status	Text size is fine. Text is fine.	Text size is not fine. Text is fine	Text size is fine. Text is not fine.	Text size is not fine. Text is not fine
Solution	There is not change.	Text size should be 14	Text should be all capital.	Text size should be 14 and it should be capital.

A. Test Case for English-Like Language Script Type

The above table described a button evaluation tast case for English-like language script type button text which is illustrated in *Table 9- 1* (appendix A).

	Test Cases			
	Case 1	***Case 2***	***Case 3***	***Case 4***
Language script type	Tall	Tall	Tall	Tall
Curent text size	15	!=15	15	!=15
Button name	All caps	All caps	Not Caps	Not Caps
Expected Status	Text size is fine. Text is fine .	Text size is not fine. Text is fine.	Text size is fine. Text is not fine.	Text size is not fine. Text is not fine.
Solution	There is not change.	Text size should be 15	Text should be all capital.	Text size should be 15 and text should be capital.

B. Test Case for Tall Language Screept Type

The above table illustrated button evaluation test cases for Tall language script type such as Hindi, Bengali and etc. This language information's are described in *Table 9- 1* (appendix A).

	Case 1	***Case 2***	***Case 3***	***Case 4***
Language script type	Dense	Dense	Dense	Dense
Curent text size	15	!=15	15	!=15
Button name	Bold	Bold	Not bold	Not bold
Expected Status	Text size is fine. Text style is fine .	Text size is not fine. Text style is fine .	Text size is fine. Text is not fine.	Text size is not fine. Text is not fine .
Solution	There is not change.	Text size should be 15.	Text should be bold.	Text size should be 15 and text style should be bold .

C. Test Case for Dense Language Screept Type

The above table defined button evaluation test case for Dense language script type such as Japanese, Korean, and Chinese languages which are defined in *Table 9- 1* (appendix A).

The multi-language button evaluation should be performed on different language user interface integrated with the proposed button typography evaluation framwork and finaly the evalution results are compared with it's the above test case.

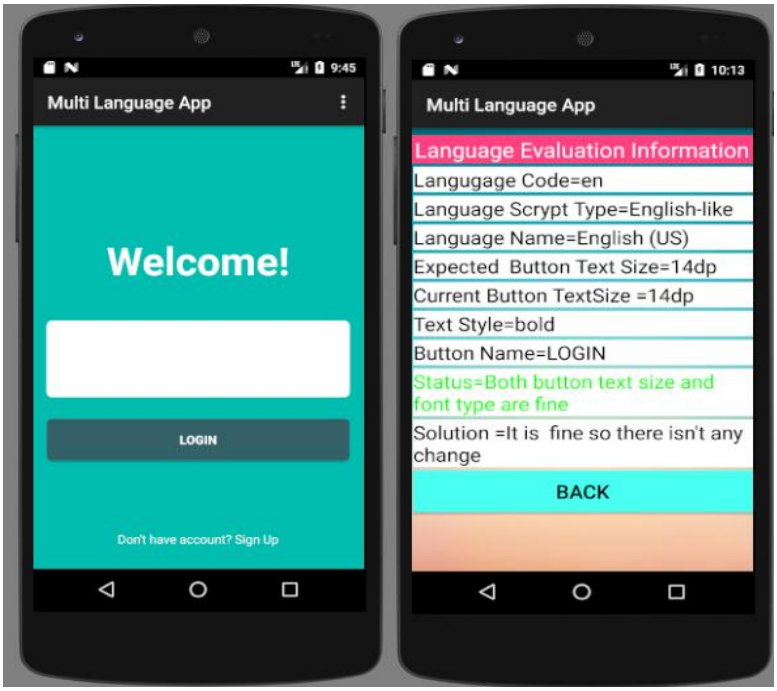


Figure 7:3 Button Evaluation English Language Host Application and Result

The above diagrams are described button evaluation for English (US) language. This language grouped under English-like language script type. According to the test case button text should be 14 dp and the text should be all capital letter. The evaluation can fulfill this test case. Additionally, Hindi user interface button evaluation result is descried in *Figure 9: 3* (appendix C). Generally, the proposed framework was evaluated on seven different language user interface and recored the result in table below.

According to below evaluation results the proposed framework can evaluate button based on the language scrip type and it provided recommendation solution for button design problems based on the button typography guidelines.

Table 7-6 Button Evaluation on Different Language Results

	Language code	***Language script type***	***Language name***	***Current button size (dp)***	***Expected button size(dp)***	***Text style***	***Button name***	***Status***	***Solution***
1	en	English-like	English-like(US)	14	14	Bold	LOGIN	Both button text size and font type is fine.	There isn't any change.
2	hi	Tall	Hindi	14	15	Bold	लॉगिन	Button text size and font type aren't fine.	Button text size should be 15 dp and button text should be all capital
3	fr	English-like	French (European)	14	14	Bold	connexion	Button text size is fine but font type isn't fine.	Button text should be all capital.
4	ru	English-like	Russian	15	14	Normal	Русский	Button text size and font type aren't fine.	Button text size should be 14 dp and button text should be all capital
5	de	English-like	German	14	14	Normal	DEUTSCH	Both button text size and font type is fine.	There isn't any change.
6	ko	Dense	Korean	15	15	Bold	로그인	Both button text size and font type is fine.	There isn't any change.
7	ne	Tall	Nepali	15	15	Bold	लग - इन	Button text size is fine but font type isn't fine.	Button text should be all capital.

7.4. Line Length Evaluation Result

The proposed framework evaluated on text view and edit text user interface elements. We proposed line length evaluation test cases based on *Table 4-2* line length evaluation guidelines for English language. This proposed evaluation test cases are illustrated in tables below.

Table 7-7Short Line Evaluation Test Case

Short line	***Number of line <=15***	***numbers of characters>15 && number of characters<=30***	***Numbers of characters>30***
Number of line==1	Ideal range	Ideal range	Too wide
numbers of line <=2 && first line	Too narrow	Ideal range	Too wide
Number of line <= 2 && line2	Ideal	Ideal	Too wide

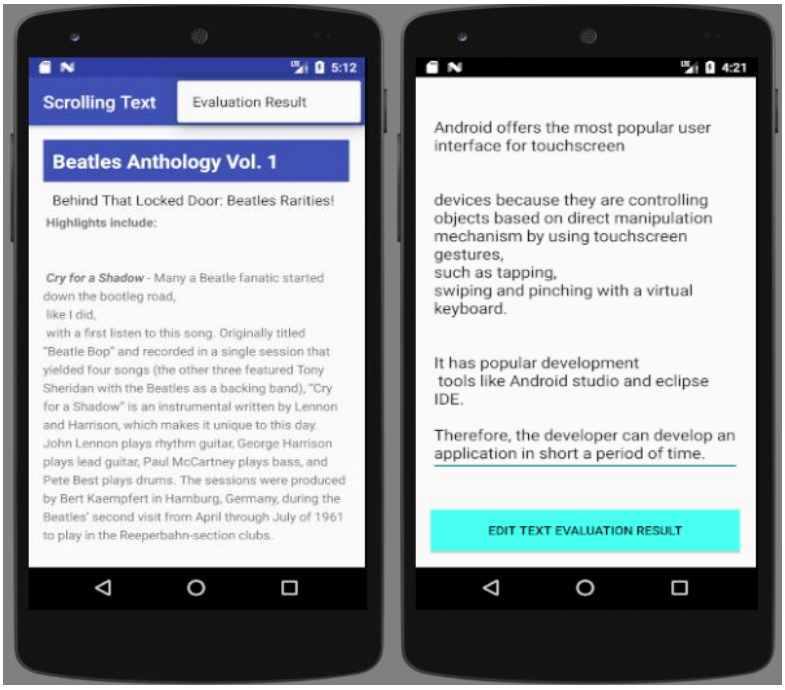
Table 7-8Text Body Evaluation Test Case

Text body	***Number of line <=30***	***numbers of characters>30 && number of characters<=60***	***Numbers of characters>60***
Number of line>2&& line !=last line	Too narrow	Ideal range	Too wide
numbers of line >2 &&line== last line	Ideal range	Ideal range	Too wide

Additionally, the proposed test case including free line (numbers of characters per line is zero) information. The host application has free line between evaluated texts, this indicates the current paragraph finished and the next paragraph started. More than one free line available in consecutively, the line is unnecessary line.

The line length evaluation framework was evaluated on scrollable text and edit text user interface elements. The scrollable text user interface downloaded from the internet. Additionally, we developed sample user interface. This user interface has one edit text and we inserted sample text

and evaluated it. The evaluated text view and edit text evaluation results with diagrams shown in below.



A. Scrollable Text View Host Application B. Edit Text Host Application

Figure 7:4 Line Lenth Evaluation Host Applications

The above scrolabe text evaluated host application has 97 line and 4074 characters . This difficalt to show all evaluation line resuts . Therefore, we descriped only the summerised evaluation results in below.

```
I/LINE INFORMATION: number of line=97    Total number of character=4074

I/Evaluation Result:  Too Narrow and Too Wide Lines
I/ Too Narrow Line: 23      Start line=831      End line=853        No Of Character 22
I/ Too Narrow Line: 24      Start line=854      End line=867        No Of Character 13
I/ Too Narrow Line: 39      Start line=1514     End line=1523       No Of Character 9
I/Unnecessary line: Must be Removed
I/ Unnecessary Lines: 21        Start line=780      End line=780        No Of Character 0
```

Figure 7:5The Scrolable Text Line Length Evaluation Result

The edit text evaluation results has the following basic information such as total numbers of line, total numbers of charaters ,and numbers of charaters per line. Addationally, the proposed framework can be generated summerized evaluation results too narrow , too wide and unnessary lines.

```
I/LINE INFORMATION: number of line=19    Total number of character=382
I/Line: 1       Start line=0        End line=36     No Of Character 36
I/Line: 2       Start line=37       End line=62     No Of Character 25
I/Line: 3       Start line=63       End line=63     No Of Character 0
I/Line: 4       Start line=64       End line=64     No Of Character 0
I/Line: 5       Start line=65       End line=101        No Of Character 36
I/Line: 6       Start line=102      End line=138        No Of Character 36
I/Line: 7       Start line=139      End line=169        No Of Character 30
I/Line: 8       Start line=170      End line=180        No Of Character 10
I/Line: 9       Start line=181      End line=198        No Of Character 17
I/Line: 10      Start line=199      End line=234        No Of Character 35
I/Line: 11      Start line=235      End line=245        No Of Character 10
I/Line: 12      Start line=246      End line=246        No Of Character 0
I/Line: 13      Start line=247      End line=247        No Of Character 0
I/Line: 14      Start line=248      End line=274        No Of Character 26
I/Line: 15      Start line=275      End line=313        No Of Character 38
I/Line: 16      Start line=314      End line=320        No Of Character 6
I/Line: 17      Start line=321      End line=321        No Of Character 0
I/Line: 18      Start line=322      End line=361        No Of Character 39
I/Line: 19      Start line=362      End line=400        No Of Character 38
I/Evaluation Result:  Too Narrow and Too Wide Lines
I/ Too Wide Line: 1     Start line=0        End line=36     No Of Character 36
I/ Too Narrow Line: 8       Start line=170      End line=180        No Of Character 10
I/ Too Narrow Line: 9       Start line=181      End line=198        No Of Character 17
I/ Too Narrow Line: 14      Start line=248      End line=274        No Of Character 26
I/ Too Wide Line: 18        Start line=322      End line=361        No Of Character 39
I/ Too Wide Line: 19        Start line=362      End line=400        No Of Character 38
I/Unnecessary line: Must be Removed
I/ Unnecessary Lines: 4     Start line=64       End line=64     No Of Character 0
I/ Unnecessary Lines: 13        Start line=247      End line=247        No Of Character 0
```

Figure 7:6The Edit Text Line Length Evaluation Result

Acording to the above evaluation results , the line length evaluation framework can extract too narow, too wide and unnecessary lines based on the above test cases. However, the proposed framework can be identify the paragraph only get freeline. .But, the paragraph may start without free line. This is the limitation of the line length evaluation framework.

7.5. Margin Evaluation Result

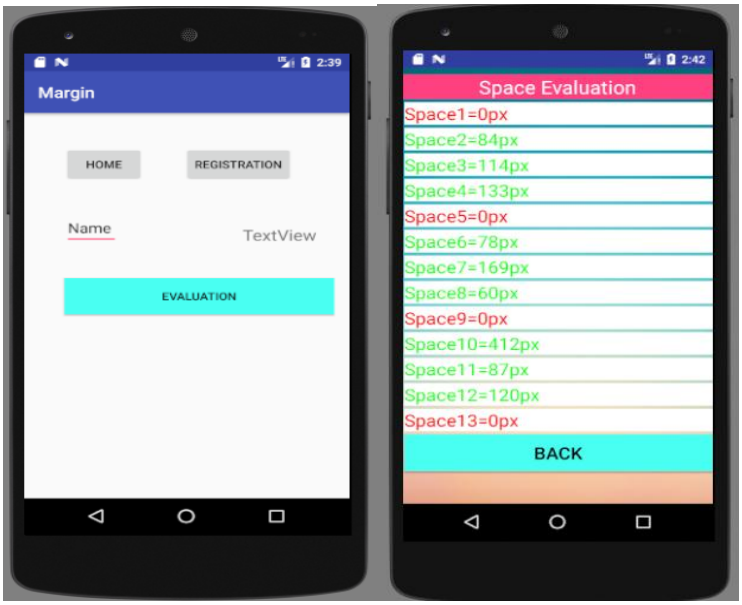


Figure 7:7 Margin Evaluation Host Application and Its Result

The margin evaluation framework is evaluated on different user interface element and generated evaluation results based on the given guidelines. The evaluation results illustrated in below diagrams. The above margin evaluation host application user interface evaluation result is described in figure below.

```
I/Summerized: User Interface Element Evaluation Result
I/Expected: Left Or Right space=11          Top Or Bottom space=13
I/Size:          Layout      Total Element       Free Layout
I/Width:         1080        525.0               555.0
I/Height:        1536        252.0               1284.0
I/User Interface:                                Position                                Margin
I/Element Name:      Width   Height   Left   Top  right   Bottom     Left    Top      right   Bottom
I/Home:              264     144      135    132     399     276      0      84       0       0
I/Registration:      362     144      532    132     894     276      0      0        78      0
I/Name:              180     136      135    390     315     526      87     114      0       120
I/TextView:          245     81       727    445     972     526      0      0        60      0
I/Space:         space Type      Current Value(px)   Minimum Expected Value(px)   Status
I/Space 1:       Horizontal      0                   11                          It is not fine.
I/Space 2:       Vertical        84                  13                          It is fine.
I/Space 3:       Vertical        114                 13                          It is fine.
I/Space 4:       Horizontal      133                 11                          It is fine.
I/Space 5:       Horizontal      0                   11                          It is not fine.
I/Space 6:       Horizontal      78                  11                          It is fine.
I/Space 7:       Vertical        169                 13                          It is fine.
I/Space 8:       Horizontal      60                  11                          It is fine.
I/Space 9:       Horizontal      0                   11                          It is not fine.
I/Space 10:      Horizontal      412                 11                          It is fine.
I/Space 11:      Horizontal      87                  11                          It is fine.
I/Space 12:      Vertical        120                 13                          It is fine.
I/Space 13: is not available
I/Summarized: Space Calculation Information
I/space1=: HomeLeftMargin                          space2=HomeToptMargin
I/space3=: NameTop - HomeBottom                       space4=RegistrationLeft - HomeRight
I/space5=: RegistrationTopMargin                          space6=RegistrationRightMargin
I/space7=: TextViewTop - RegistrationBottom                  space8=TextViewRightMargin
I/space9=: TextViewBottomMargin                        space10=TextViewLeft - NameRight
I/space11=: NameLeftMargin                          space12=NameBottomMargin
```

Figure 7:8Margin Evaluation Result

The above evaluation results compared with manual expected free space result in below.

Table 7-9 Total Width and Height Results

	Value	***Alignment***	***Element***	***Total width***
Layout width (px)	1080	Horizontal one	Home	626
Layout height (px)	1536		Registration	
Home width	264	Horizontal two	Name	426
Home height	144		Text view	
Registration width	362			Total height
Registration height	144	Vertical one	Home	310
Name width	180		Name	
Name height	166	Vertical two	Registration	389
Text view width	245		Text view	
Text view height	81			

Table 7-10 Manual Minimum Expected Free Space Results

Results	
Free width one	1080-626=454
Free width two	1080-426=654
Free height one	1536-310=1226
Free height two	1536-389=1147
Expected space left and right one	451*0.02=9.02
Expected space left and right two	654*0.02=13.08
Expected space top and bottom one	1226*0.01=12.26
Expected space top and bottom two	1147*0.01=11.47

Table 7-11Manual verses Proposed Expected Free Space Result

Expected free space calculated	***Left and right***		***Top and bottom***	
	Horizontal one	***Horizontal two***	***Vertical one***	***Vertical two***
Actual(manual)	9.02	13.08	12.26	11.47
Proposed framework	11		13	
Variation	-1.8	2.8	-0.74	-1.53

Similar to the above margin evaluation process, we are evaluated three different user interfaces. These user interface elements have aligned in different ways: such as two vertical and three horizontal alignments, all element are aligned vertically and horizontally respectively. Each detail evaluation process and results are illustrated in *Figure 9: 4*, *Table* 9- 20, *Table 9- 21*, and *Table 9- 22* (appendix C). However, the summarized manual and automated evaluation method comparison results are shown in table below. The variation result has both negative and positive values but there is not affected the evaluation values. According to the below comparison results the automated expected free space calculation method has approximation values with manual method (minimum variation) except 4.99. When evaluated elements have similar size and equal

numbers of elements are aligned in the same direction (horizontally or vertically), the automated and manual calculation method have similar results.

Table 7-12Manual verses Automated Expected Free Space Results

Alignment	***Comparison***		
Aligned Two Columns and three rows	***Manual method***	***Automated method***	***Variation***
Horizontal one	8.01	13	-4.99
Horizontal two	14.98		1.98
Horizontal three	14.70		1.70
Vertical one	13.92	12	1.92
Vertical two	11.04		-0.96
All are aligned vertical			
Horizontal one	14.40	15	-0.60
Horizontal two	14.50		-0.50
Horizontal three	14.22		-0.88
Horizontal four	15.00		0.00
Vertical one	9.60	10	-0.60
All are aligned horizontally			
Horizontal one	4.62	5	-0.32
Vertical one	13.80	14	-0.20
Vertical two	13.62		-0.38
Vertical three	13.44		-0.66
Vertical four	13.26		-0.74

7.6. Action Bar Menu Item Evaluation Result

The Action bar menu item evaluation framework evaluated numbers of menu item on the action bar. We proposed test case to evaluate the proposed framework based on *Table 4-3* that described in chapter four. This test case is illustrated in below table.

Table 7-13Action Bar Menu Item Evaluation Test case

	Screen width >360	***Scree width >= 360 && Scree width<=499***	***Scree width >= 500&& Scree width<=599***	***Scree width >=600***
Numbers of menu items<=2	It is fine	It is fine	Its fine	It is fine
Numbers of menu items==3	It is not fine Over item=1	It is fine	It is fine	It is fine
Numbers of menu items==4	It is not fine Over item=2	It is not fine Over item=1	It is fine	It is fine
Numbers of menu items==5	It is not fine Over item=3	It is not fine Over item=2	It is not fine Over item=1	It is fine
Numbers of menu items>5	It is not fine Over item=number of items-2	It is not fine Over item=number of items-3	It is not fine Over item=number of items-4	It is not fine Over item=number of items-5

The proposed framework was evaluated on different host applications which have action bar menu items. These host application has different numbers of menu items on the action bar.

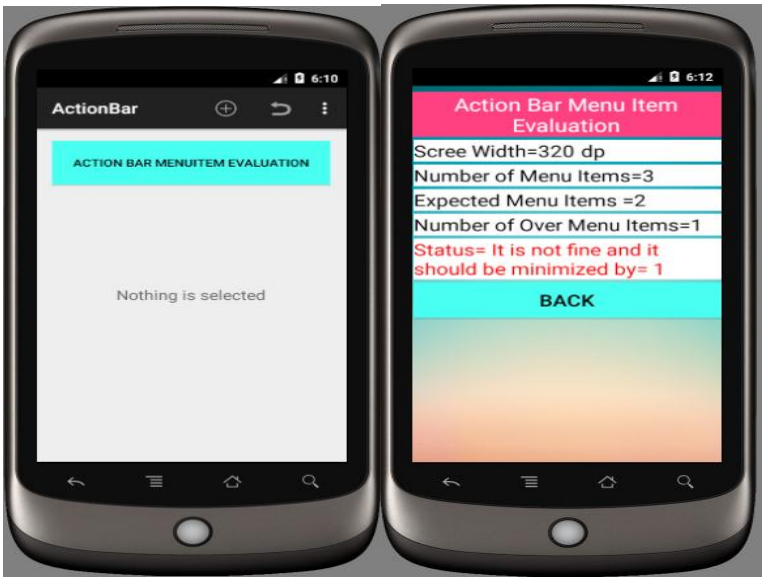


Figure 7:9 Action Bar Menu Evaluation Host Application And Evaluation Result

We are described two sample action bar menu item evaluation results with the above figure and *Figure 9: 6* (appendix C). The above figure is shown the host application and evaluated result. This mobile device has 360 dp width and the host application has three menu item on the action bar. According to the above test case, this mobile device should be maximum two menu items, but the action bar has three menu items. Therefore, the proposed framework displayed recommendation solution.

Table 7-14 Action Bar Menu Item Evaluation Results

	Screen width(dp)	***No of menu item***	***Expected menu item***	***No of over menu item***	***Status***
1	360	4	3	1	It is not fine and it should be minimized by 1
2	320	4	2	2	It is not fine and it should be minimized by 2
3	411	4	3	1	It is not fine and it should be minimized by 1
4	1280	4	5	-1	It is fine
5	360	3	3	0	It is fine
6	320	3	2	1	It is not fine and it should be minimized by 1
7	411	3	3	0	It is fine
8	1280	3	5	-2	It is fine

The proposed Action bar evaluation performed on different mobile device and user interface and recorded the result in the above table. All evaluation results are shown that proposed evaluation framework can be evaluated based on the guidelines and conformance the test cases. According to the above action bar evaluation result, the proposed framework can be evaluated action bar menu item based on the guidelines which illustrated in *Table 4-3* in chapter four.

CHAPTER EIGHT

8. Conclusion and Future Work

In this chapter we descried the summary of the proposed automated user interface usability evaluation framework and future research directions.

8.1. Conclusion

Today, there are different types of smart phone device available in the market. Each smart phone has their OS and application features. Android is the popular software stack for smartphone device which includes an operating system, middleware, and key applications. Design appropriate user interface is the critical issue for smart phone device because smart phone provide various type of services but it has limited screen size and touch sensitive devices Hence, different researchers proposed automated usability evaluation framework and tool for mobile application specially, Android and iOS. But, most proposed frameworks are evaluated limited user interface elements and dynamic aspects of user interface. Additionally, user interface design guidelines are very important document for design appropriate user interface components in smart phone application development. However, all Android mobile user interface design guidelines are not organized in proper way. Therefore, the developer can difficult to apply the application design guidelines.

This study investigated an automated user interface usability evaluation framework for Android application based on Android user interface design guidelines. Several steps and procedures are applied to propose the framework. In the beginning it focuses on selecting an important user interface elements and collecting their qualitative metrics from Android application design guideline documents and Android websites. Then, proposed the automated user interface usability evaluation framework. This framework consists of various types of user interface element which are descried in *chapter four*. Specifically, the proposed prototype has five user interface element evaluation and mobile device evaluation frameworks. Such as mobile device evaluation, header and footer evaluation, line length evaluation, button typography evaluation, margin evaluation and action bar menu item evaluation frameworks.

The proposed framework can provide user interface analysis results, current status and critiques (recommendation) to the developer. The proposed mobile device framework offered twofold advantage. First, the evaluation result used to as input for user interface element evaluation like screen width, screen density, screen height, OS version. Second, gathering mobile device information for farther knowledge. The header and footer evaluation framework can be evaluated the host application header and footer height based on mobile screen height. This framework provided header and footer status. When header and footer height greater than 30% of mobile screen height then it provided numbers of dp minimized from the header and/or footer height. The multi-language button evaluation framework can be evaluated button based on language script type and provided recommended solution for design defects. Additionally, the multi-language evaluation framework provides button information like text size, text style, and language type information based on language script type. The line length evaluation framework can be evaluated text size of user interface component like edit text and text view.

Basically, this evaluation framework can be extracted too narrow, too wide and unnecessary lines from evaluation text. Additionally, the line length evaluation framework provides line information such as total number of characters, total number of line, start and end lines, number of characters per line. Margin evaluation framework is a part of automated user interface usability evaluation framework which evaluated the space between user interface elements. Basically, this framework provides the current user interface margin values, the space between user interface elements, expected minimum free space between user interface value for left, top, right, and bottom space for each evaluated user interface elements. The action bar menu item evaluation framework can evaluate number of menu items based on mobile screen size. This evaluation framework provides current menu item, expected menu item and over numbers of menu items.

Generally, the proposed frameworks can minimize some user interface design problems like header and footer height, button design based on language type, line length and action bar menu item. They can also minimize Android user interface design problems before releasing them in to the market. Additionally, this evaluation framework evaluated user interface without any user behavior interference, without involvement of user and tester. Hence, the proposed framework can minimized evaluation cost and time and human power.

However, the automated user interface usability evaluation framework has the following limitations. The line length evaluation framework can identified short line and text body based on free line between each paragraph. But, the text may be started the paragraph without free line. The margin evaluation framework complexity increase as evaluation user interface element increase to implement the framework. The proposed framework may increase the developer task during integrated with the host application. The action bar menu evaluation framework can be evaluated total number of menu item on the action bar rather than classified into show menu item and overflow menu item.

8.2. Recommendation

User interface design guidelines are important to develop appropriate user interface. Hence, we recommended to Android mobile application developer to design mobile application based on user interface design guidelines. This detects and minimize design problems before realize the application on the market.

8.3. Future Work

Due to the limited time and resources during the course of this thesis, it was not possible to implement all the user interface elements in this study and further improvement of the proposed framework may require. Some of them are implemented; however, several user interface elements require further research:

The proposed framework need enhanced to achieve the following features:

- The proposed framework need to be implement and applicable in real Android user interface Development Company.
- The automated user interface evaluation framework for Android application integrated without source code modification by Aspect oriented method.
- The line length evaluation framework can identify short and text body without free line.
- Proposed margin evaluation framework should be evaluated without affected on numbers of evaluated user interface element.
- Additionally, study on other user interface elements like list view, check box, and radio button, application icons, and status bar user interface elements.

References

[1] shahid, "Mobile Application Development - An Insight," 12 10 2014. [Online]. Available: https://www.devsaran.com/blog/mobile-application-development-insight. [Accessed 04 06 2017].

[2] K. Divya and S. Venkata KrishnaKumar , "COMPARATIVE ANALYSIS OF SMART PHONE OPERATING SYSTEMS ANDROID, APPLE iOS AND WINDOWS," *International Journal of Scientific Engineering and Applied Science (IJSEAS)*, vol. 2, no. 2, pp. 432-438, February 2016.

[3] M. Adnan, "Usability Evaluation of Smart Phone Application Store," 25 02 2015. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:796049/FULLTEXT01.pdf.

[4] F. Lettner and C. Holzmann, "Sensing mobile phone interaction in the field," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, 2012.

[5] J. Earthy, B. S. Jones and N. Bevan, "Iso standards for user-centered design and specification of usability," *Usability in government systems: User experience design for citizens and public servants*, p. 407, 2012.

[6] W. Kluth, K.-H. Krempels and C. Samsel, "Automated Usability Testing for Mobile Applications.," in *WEBIST (2)*, 2014.

[7] X. Ma, B. Yan, G. Chen, C. Zhang, K. Huang and J. Drury, "A toolkit for usability testing of mobile applications," in *International Conference on Mobile Computing, Applications, and Services*, 2011.

[8] F. Lettner and C. Holzmann, "Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications," in *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*, 2012.

[9] D. Bader, "A framework for remote usability evaluation on mobile devices," in *Technische Universität München*, February 28, 2011.

[10] F. Balagtas-Fernandez and H. Hussmann, "A methodology and framework to simplify usability analysis of mobile applications," in *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, 2009.

[11] F. Lettner and C. Holzmann, "Usability Evaluation Framework Automated Interface Analysis for Android Applications," *Springer-Verlag Berlin Heidelberg*, vol. part II, no. EUROCAST 2011, p. 560–567, 2012.

[12] S. Baker, F. Au, G. Dobbie and I. Warren, "Automated usability testing using HUI Analyzer," in *Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on*, 2008.

[13] J. Šebek and K. Richta, "Aspect-oriented user interface design for Android applications," 2014.

[14] F. T. W. Au, S. Baker, I. Warren and G. Dobbie, "Automated usability testing framework," in *Proceedings of the ninth conference on Australasian user interface-Volume 76*, 2008.

[15] A. Beckley, "android design guidelines," February 2011. [Online]. Available: https://tctechcrunch2011.files.wordpress.com/2011/04/mm_android_design_guidelines.pdf.

[16] "Application Design and Development Guidelines," February, 2016. [Online]. Available: http://docs.kony.com/7_0_PDFs/visualizer/app_design_dev.pdf.

[17] P. Chandnani and R. Wadhvani, "Evolution of android and its impact on mobile application development," *International Journal of Scientific Engineering and Technology*, vol. 1, pp. 80-85, 2012.

[18] O. O. Okediran, O. T. Arulogun, R. A. Ganiyu and C. A. Oyeleye, "Mobile operating systems and application development platforms: A survey," *International Journal of Advanced Networking and Applications*, vol. 6, no. 1, pp. 2195-2201, 2014.

[19] A. Ahmed, A. Raza and S. Sadik, "User's Perspective of Smartphone Platforms Usability: An Empirical Study," in *Intelligent Systems, Modelling and Simulation (ISMS), 2014 5th International Conference on*, 2014.

[20] A. A. Sheikh*, P. T. Ganai**, N. A. Malik*** and K. A. Dar****, "Smartphone: Android Vs IOS," *The SIJ Transactions on Computer Science Engineering & its Applications (CSEA)*, vol. 1, no. 4, pp. 141-148, September-October 2013.

[21] C. Ziegler, "Microsoft talks Windows Phone 7 Series development ahead of GDC: Silverlight, XNA, and no backward compatibility," *Engadget, AOL*, 2010.

[22] T. Renner, "Mobile OS-Features, Concepts and Challenges for Enterprise Environments," *SNET Project Technische Universitat Berlin*, 2014.

[23] H. Sten and G. Tiger, "Web Operating System for Modern Smartphones," Chalmers University of Technology, 2011.

[24] D. Gavalas and D. Economou, "Development platforms for mobile applications: Status and trends," *IEEE software,* vol. 28, pp. 77-86, 2011.

[25] A. Niknejad, "A Quality Evaluation of an Android Smartphone Application," MS thesis, 2011.

[26] "Android operating system framework architecture images," google, [Online]. Available: https://www.google.co.uk/search?q=Android+operating+system+framework. [Accessed 30 11 2017].

[27] Nikhil M. Dongre, Tejas S. Agrawal, Ass.prof. Sagar D. Pande, "A Research On Android Technology With New Version," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 19, no. 2, pp. 65-77, Mar.-Apr. 2017.

[28] N. Bevan, J. Carter and S. Harker, "ISO 9241-11 revised: What have we learnt about usability since 1998?," in *International Conference on Human-Computer Interaction*, 2015.

[29] R. Zhou and A. H. S. Chan, "Using a fuzzy comprehensive evaluation method to determine product usability: A proposed theoretical framework," *Work,* vol. 56, pp. 9-19, 2017.

[30] A. Hussain and M. Kutar, "Usability metric framework for mobile phone application," *PGNet, ISBN,* vol. 2099, pp. 978-1, 2009.

[31] A. Alluri, "Usability testing of android applications," Diss. San Diego State University, 2013.

[32] S. L. Keenan, H. R. Hartson, D. G. Kafura and R. S. Schulman, "The usability problem taxonomy: A framework for classification and analysis," *Empirical Software Engineering,* vol. 4, pp. 71-104, 1999.

[33] "Supporting Multiple Screens," [Online]. Available: https://developer.android.com /guide/ practices/screens_support.html. [Accessed 23 07 2017].

[34] "Apple Inc. iOS human interface guidelines," Jan. 2011. [Online]. Available: http://developer.apple.Com / library / ios / documentation/ UserExperience/ Conceptual/ MobileHIG/ MobileHIG.pdf.

[35] M. Y. Ivory and M. A. Hearst, "The state of the art in automating usability evaluation of user interfaces," *ACM Computing Surveys (CSUR),* vol. 33, pp. 470-516, 2001.

[36] L. Hasan, A. Morris and S. Probets, "A comparison of usability evaluation methods for evaluating e-commerce websites," *Behaviour \& Information Technology,* vol. 31, no. 7, pp. 707-737, 2012.

[37] A. Holzinger, "Usability engineering methods for software developers," *Communications of the ACM,* vol. 48, pp. 71-74, 2005.

[38] L. Hasan, "The usefulness of user testing methods in identifying problems on university websites," *JISTEM-Journal of Information Systems and Technology Management,* vol. 11, no. 2, pp. 229-256, 2014.

[39] D. V. Rojatkar, G. M. Jengathe, A. B. Khairnar and S. A. Lengure., "ANDROID APPLICATION DEVLOPMENT SOFTWARE – ANDROID STUDIO AND ECLIPSE," March - 16. [Online]. Available: http://www.ijfeat.org/papers/march20163.pdf.

[40] "Tab Icons," [Online]. Available: https://developer.android.com/guide/practices/ui_guidelines/ icon_design_tab.html. [Accessed 18 08 2017].

[41] "Typography," [Online]. Available: https://material.io/guidelines/style/typography.html. [Accessed 05 09 2017].

[42] "Color Luminance," [Online]. Available: http://www.workwithcolor.com/color-luminance-2233.htm . [Accessed 06 09 2017].

[43] "Action Bar," [Online]. Available: https://developer.android.com/design/patterns/actionbar.html. [Accessed 29 07 2017].

[44] "Menus," [Online]. Available: https://material.io/guidelines/components/menus.html. [Accessed 28 07 2017].

[45] "Types of tabs," [Online]. Available: https://material.io/guidelines/components/tabs.html#tabs-types-of-tabs. [Accessed 02 09 2017].

[46] "Buttons," [Online]. Available: https://material.io/guidelines/components/buttons.html. [Accessed 03 09 2017].

[47] "Buttons: Floating Action Button," [Online]. Available: https://material.io/guidelines/components/ buttons-floating-action-button.html. [Accessed 24 07 2017].

[48] "Icon Design Guidelines," [Online]. Available: https://developer.android.com/guide/practices/ ui_guidelines/icon_design.html. [Accessed 02 08 2017].

[49] "Menu Icons," [Online]. Available: https://developer.android.com/guide/practices/ui_guidelines/ icon_design_menu.html. [Accessed 07 08 2017].

[50] "Action Bar Icons," [Online]. Available: https://developer.android.com/guide/practices/ui_guidelines/icon_design_action_bar.html. [Accessed 11 08 2017].

Appendixes

Appendix A: Best Practice on Android UI Development Tables

Table 9- 1 Language Script with Their Categories **[41]**

Code	***Description***	***Category***
af	Afrikaans	English-like
am	Amharic	English-like
ar	Arabic (Modern Standard)	Tall
az	Azerbaijani	English-like
bg	Bulgarian	English-like
bn	Bengali	Tall
ca	Catalan	English-like
cs	Czech	English-like
cy	Welsh	English-like
da	Danish	English-like
de	German	English-like
el	Greek	English-like
en	English (US)	English-like
en-GB	English (UK)	English-like
es	Spanish (European)	English-like
es-419	Spanish (Latin American)	English-like
et	Estonian	English-like
eu	Basque	English-like
fa	Persian	Tall
fi	Finnish	English-like
fil	Filipino	English-like
fr	French (European)	English-like
fr-CA	French (Canadian)	English-like
gl	Galician	English-like
gu	Gujarati	Tall
hi	Hindi	Tall
hr	Croatian	English-like
hu	Hungarian	English-like

hy	Armenian	English-like
id	Indonesian	English-like
is	Icelandic	English-like
it	Italian	English-like
iw	Hebrew	English-like
ja	Japanese	Dense
ka	Georgian	English-like
kk	Kazakh	English-like
km	Khmer	Tall
kn	Kannada	Tall
ko	Korean	Dense
ky	Kirghiz	English-like
lo	Lao	Tall
lt	Lithuanian	English-like
lv	Latvian	English-like
mk	Macedonian	English-like
ml	Malayalam	Tall
mn	Mongolian	English-like
mr	Marathi	Tall
ms	Malay	English-like
my	Burmese (Myanmar)	Tall
ne	Nepali	Tall
nl	Dutch	English-like
no	Norwegian (Bokmål)	English-like
pa	Punjabi	Tall
pl	Polish	English-like
pt	Portuguese (Brazilian)	English-like
pt-PT	Portuguese (European)	English-like
ro	Romanian	English-like
ru	Russian	English-like
si	Sinhala	Tall
sk	Slovak	English-like
sl	Slovenian	English-like
sq	Albanian	English-like
sr	Serbian (Cyrillic)	English-like

sr-Latn	Serbian (Latin)	English-like
sv	Swedish	English-like
sw	Swahili	English-like
ta	Tamil	Tall
te	Telugu	Tall
th	Thai	Tall
tr	Turkish	English-like
uk	Ukrainian	English-like
ur	Urdu	Tall
uz	Uzbek	English-like
vi	Vietnamese	Tall
zh-CN	Chinese (Simplified, Mandarin)	Dense
zh-HK	Chinese (Mandarin, Hong Kong)	Dense
zh-TW	Chinese (Traditional, Mandarin)	Dense
zu	Zulu	English-like

Table 9- 2 User Interface Text Size Based on Language Type **[41]**

	Headline	***Title***	***Sub heading***	***Body1***	***Body2***	***Caption***
English and English like (sp)	Regular 24	Medium 20	Regular 16	Regular 14	Medium 14	Regular 12
Tall (sp)	Regular 24	Bold 21	Regular 17	Regular 15	Bold 15	Regular 13
Dense (sp)	Regular 24	Medium 21	Regular 17	Regular 15	Medium 15	Regular 13

Table 9- 3 Color and Contrast Ratio Based on Typeface **[41]**

	Headline	***Title***	***Subheading***	***Body 1***	***Body2***	***Caption***	***menu***	***Button***
Color	black	black	black	black	black	black	black	black
Percentage (%)	87	87	87	87	87	54	87	87

Table 9- 4 Typeface Line Height Based on Language Type **[41]**

		Headline	***Sub headlining 1***	***Sub headline 2***	***Body 1***	***Body 2***
English and English like Language scripts	Type (sp)	24	15 and 16	16	13 and 14	13 and 14
	Line height (Leading dp)	32	24	28	20	24
Dense and tall language scripts	Type (dp)	24	16 and 17	16 and 17	14 and 15	14 and 15
	Leading (dp)	34	26	30	23	26

Table 9- 5 Menu Item Height Based on Screen Density **[15]**

Screen densities	***ldpi***	***mdpi***	***hdpi***
Consistent Height	50x	66x	100x

Table 9- 6 The Alignment and Font Size of Simple Menu Items **[44]**

Alignment	***Menu item height***	***Menu item text left padding***	***Menu item text bottom padding***	***Top and bottom padding***	***Text height***
Density values	48 dp	16dp	20dp	8dp	16sp

Table 9- 7Android Tab Bar Sizes Based on Mobile Screen Densities **[15]**

Screen densities	***ldpi***	***mdpi***	***hdpi***
Tab bar sizes	240x48	320x64	480x96

Table 9- 8 Maximum and Minimum Width for Fixed Tab Design **[45]**

	Maximum width	***Minimum width***	
		Larger view	**Smaller view**
Tab Width	264 dp	160 dp	72 dp
Alignment	-	Centered	Full- width

Table 9- 9Tab Padding and Indicator for Fixed Tab **[45]**

	Height	***Padding***			***Indicator***	
		left and right of text	***From bottom***		***Height***	***Color***
			Singles line of text	Two line of text		#fff or accent color
Size in dp	48	12	20	12	2	

Table 9- 10 Fixed Tab Design Information **[45]**

	Tabs with icon and text	***Tabs with icon only***	***Tabs with text only***
Height	72 dp	48 dp	14 sp Roboto medium All cups Text is Centered horizontally and vertically Active color :#fff accent color Unfocused tab color :#fff 70%
icon	24x24 dp	24x24 dp	
Content alignment	Centered horizontally	Centered horizontally and vertically	
Padding	10 dp between icon and text 16 dp under text	12 dp under icon	

Table 9- 11 Button Location on Mobile Screen **[46]**

	Button alignment		
	On the screen	**The affirmative button**	**Dismissive button**
Standard dialogs	Right	On the right	On the left
Form	Left	On the Left	On the right

Table 9- 12 Menu Icon Size for Android 2.3 and Later Versions **[49] [15]**

Mobile screen densities	***Menu Item dimension***		
Screen size (Resolution)	***Full Asset***	***Icon***	***Square Icon***
Low density screen(ldpi)	36 x 36 px	24 x 24 px	22 x 22 px
Medium density screen(mdpi)	48 x 48 px	32 x 32 px	30 x 30 px
High density screen(hdpi)	72 x 72 px	48 x 48 px	44 x 44 px

Table 9- 13 Menu Icon Information for Android 2.3 and Later Versions **[49] [15]**

Corner rounding		***Gradient***	***Inner shadow***	***Inner bevel***
Screen resolution	Corner radius	90°, from #8C8C8C to #B2B2B2	#000000,20% opacity angle 90° distance 2 px size 2 px	depth 1% direction down size 0 px angle 90° altitude 10° highlight #ffffff, 70% opacity shadow #000000, 25% opacity
ldpi	3px			
mdpi	2 px			
hdpi	1.5 px			

Table 9- 14 Menu Icon Information for Android 2.2 and Earlier Versions **[49] [15]**

Screen resolution	***Icon size in pixels***	***Safe frame pixel***
ldpi	4X24	3px
mdpi	32X32	4px
hdpi	48X482	6px

Table 9- 15 Menu Icons Color Plate for Android 2.2 and Earlier Versions **[49] [15]**

Color	***Color combination***	***Advantage***
White	r 255|g 255| b 255	Used to outer glow and bevel highlight
Fill gradient	1: r 163 | g 163 | b 163 2: r 120 | g 120 | b 120	Used as color fill.
Black	r 0 | g 0 | b 0	Used to inner shadow and bevel shadow

Table 9- 16 Action bar Icon Design based on Screen Densities

	ldpi	***mdpi***	***hdpi***	***xhdpi***
Action bar icon size	18x18 px	24x24px	36x36px	48x48 px

Table 9- 17 Tab Icons Information for Android 1.6 and Earlier Versions **[40]**

Tab icon Information	***Font part***	***Inner shadow***	***Inner bevel***
Unselected tab icons	Gradient overlay | angle 90^0 Bottom color: r223|g223|b223 Top color : r249|g249|b249 Bottom color location: 0% Top color location : 75%	Black|10%opacity| angle 90^0 distance 2px| size 2px	Depth 1% |direction down |size 0px|angle 90^0 | altitude 10^0 highlight white 70% opacity shadow black 25% opacity
Selected tab icons	Use fill gradient from color palette	Black|20% opacity| angle 900 |distance 2px |size 2px	Depth 1% |direction down |size 0px |angle 90^0| altitude 10^0 highlight white 70% opacity shadow black 25% opacity

Table 9- 18Tab Icon Dimension for Android 2.0 up to 2.3 Versions **[40]**

Tab Bar Icon Dimension	***Screen Density***		
	ldpi	***mdpi***	***hdpi***
Full Asset	24x14 px	32 x 32 px	48x 48 px
Icon	22 x 22 px	28 x 28px	42 x 42px

Table 9- 19Tap Icon Information for Android 2.0 up to 2.3 Versions **[40]**

Style and affects for	***Fill color***	***Outer glow***
un selected tab icons	#808080	
selected tab icons	#FFFFF	#000000,25% opacity, size 3px

Appendix B: Prototype Implementation Sample Diagram

The next figure illustrated the margin evaluation implementation for four user interface elements when all aligned vertically.

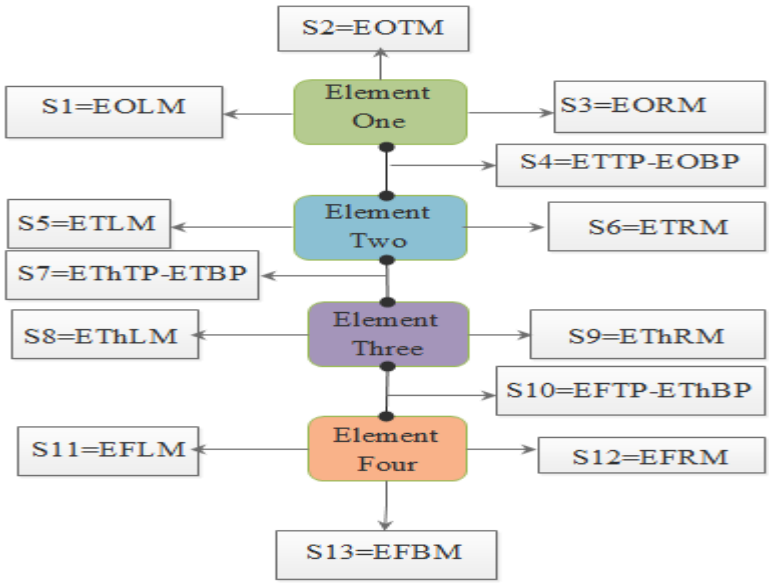


Figure 9: 1 Calculate Space between User Elements When All Are Aligned Vertically

Appendix C: Prototype Evaluation Sample Results

The proposed framework prototype evaluation sample results are descried in below diagrams and tables.

Figure 9: 2 Heard and Footer Evaluation Host Application and Result

The above diagrams are described header and footer evaluation host application and the evaluation results. The proposed framework generates evaluation results and recommendation. When host application has greater than expected header and footer height, the proposed framework display recommendation solution and the status display by red color.

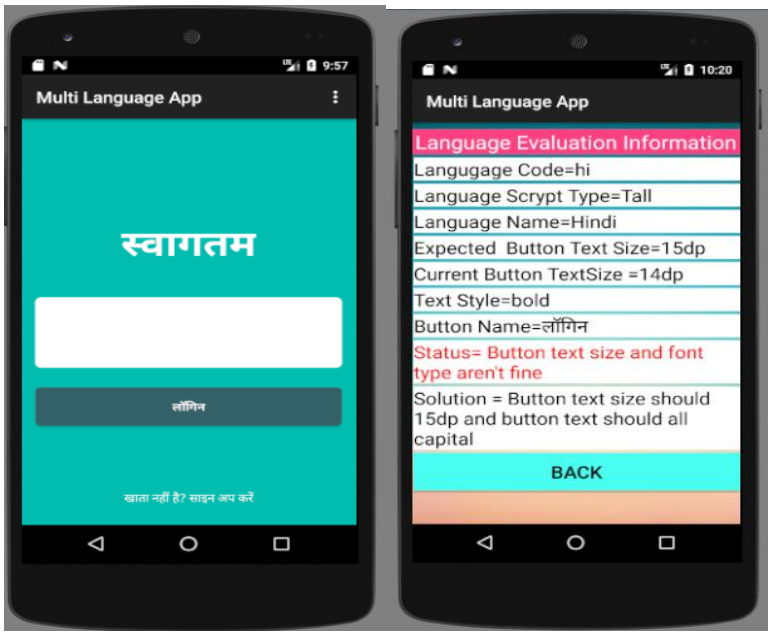


Figure 9: 3Button Evaluations for Hindi Language Host Application and Its Result

The above diagrams are described the button evaluation for Hindi language host application and the evaluation result. According to language script type, Hindi is a type of Tall language script type. We evaluated the result based on Tall language script test case. The evaluation result conformance result with the Tall language test case.

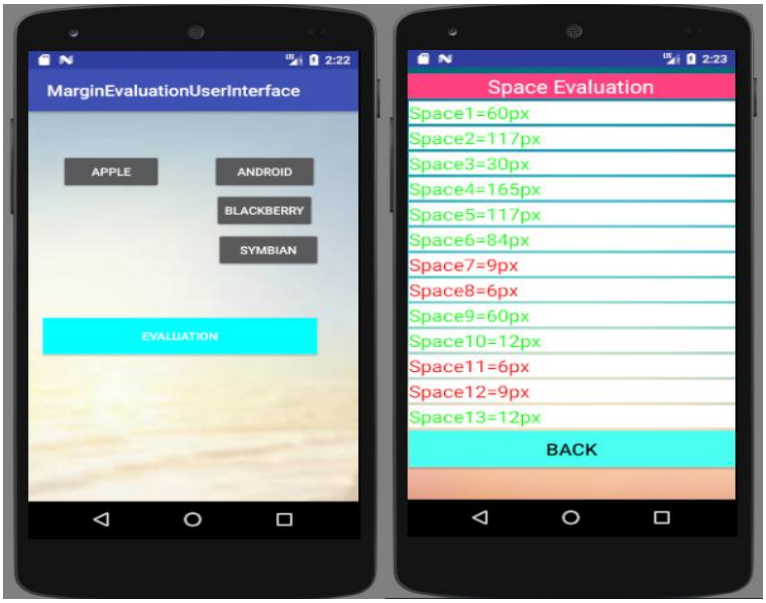


Figure 9: 4 Margin Evaluation User Interface Two

In below diagrams are illustrated user interface element margin evaluation and its result when user interface elements are aligned two vertically and three horizontally.

```
I/Expected: Left Or Right space=13          Top Or Bottom space=12
I/Size:         Layout      Total Element       Free Layout
I/Width:        1080        450.0               630.0
I/Height:       1536        288.0               1248.0
I/User Interface:                               Position                            Margin
I/Element Name:     Width   Height   Left   Top  right   Bottom     Left    Top      right   Bottom
I/Android:          345     144     603     165     948     309      0      117     84      6
I/Apple:            330     144     108     165     438     309     60      117     0       30
I/BlackBerry:       331     144     609     318     940     462     6       3       60      6
I/Symbian:          345     144     615     474     960     618     6       6       9       12

I/Space:        space Type      Current Value(px)   Minimum Expected Value(px)   Status
I/Space 1:      Horizontal      60                  13                      It is fine.
I/Space 2:      Vertical        117                 12                      It is fine.
I/Space 3:      Vertical        30                  12                      It is fine.
I/Space 4:      Horizontal      165                 13                      It is fine.
I/Space 5:      Vertical        117                 12                      It is fine.
I/Space 6:      Horizontal      84                  13                      It is fine.
I/Space 7:      Vertical        9                   12                      It is not fine.
I/Space 8:      Horizontal      6                   13                      It is not fine.
I/Space 9:      Horizontal      60                  13                      It is fine.
I/Space 10:     Vertical        12                  12                      It is fine.
I/Space 11:     Horizontal      6                   13                      It is not fine.
I/Space 12:     Horizontal      9                   13                      It is not fine.
I/Space 13:     Vertical        12              12                      It is fine.
I/Summarized: Space Calculation Information
I/space1=: AppleLeftMargin                          space2=AppleTopMargin
I/space3=: AppleBottomMargin                        space4=AndroidLeft - AppleRight
I/space5=: AndroidToptMargin                        space6=AndroidRightMargin
I/space7=: BlackBerryTop - AppleBottom                  space8=BlackBerryLeftMargin
I/space9=: BlackBerryRightMargin                    space10=SymbianTop - BlackBerryBottom
I/space11=: SymbianLeftMargin                       space12=SymbianRightMargin
I/space13=: SymbianBottomMargin
```

Figure 9: 5Margin Evaluation Result User Interface Two

The above results user interface margin and free space evaluation results between elements. According to the above margin information, we calculated each user interface elements margin free space values manually and it compared with the above automated evaluation results. The summarized final results are descried in below tables

Table 9- 20Margin Evolution Information when aligned vertically and horizontally

Element Name	***Size***		***Position***				***Margin***			
	Width	***height***	***Left***	***Top***	***Right***	***Bottom***	***Left***	***Top***	***Right***	***Bottom***
ANDROID	345	144	603	165	948	309	0	117	84	6
APPLE	330	144	108	165	438	309	60	117	0	30
BLACKBERRY	331	144	609	318	940	462	6	3	60	6
SYMBIAN	345	144	615	474	960	618	6	6	9	12

A. User interface margin evaluation element information

Alignment	***Element name***	***Total width***	***Free Layout width***	***Expected free space***
Horizontal one	APPLE	675	1080-675 =405	8.01
	ANDROID			
Horizontal two	BLACKBERRY	331	1080-331=479	14.98
Horizontal three	SYMBIAN	345	1080-345=7.35	14.7
		Total height	***Free Layout height***	
Vertical one	APPLE	144	1536-144=1392	13.92
Vertical two	ANDROID	432	1536-432=1104	11.04
	BLACKBERRY			
	SYMBIAN			

B. Margin Evaluation Results by Manual Calculation Method

Similar to the above margin evaluation process, we evaluated two user interfaces which are consisted four user interface elements. These user interface element information's are illustrated in table below that are aligned vertically and horizontally respectively.

Table 9- 21 Margin Evolution Information when Elements are aligned vertically

Element Name	***Size***		***Position***				***Margin***			
	Width	***Height***	***Left***	***Top***	***Right***	***Bottom***	***Left***	***Top***	***Right***	***Bottom***
REGISTRATION	360	144	93	63	453	207	45	15	45	6
UPDATE	345	144	93	213	438	357	0	0	9	9
DELET	369	144	93	405	462	549	0	39	90	0
CLEAR	330	144	93	624	423	768	0	75	9	0

A. User Interface Elements Information

The above *Table 9- 21* A illustrated margin evaluation element information for all evaluated user interface element aligned vertically.

Alignment	***Element Name***	***Total Width***	***Free Layout Width***	***Expected Free Space***
Horizontal one	REGISTRATION	360	1080-360 =720	14.4
Horizontal two	UPDATE	355	1080-355=725	14.5
Horizontal three	DELET	369	1080-369=711	14.22
Horizontal four	CLEAR	330	1080-330=750	15
		Total Height	***Free Layout Height***	
Vertical one	REGISTRATION	576	1536-576=960	9.6
	UPDATE			
	DELET			
	CLEAR			

B. Total Size, Free and Expected space Results

The above Table *9- 21* B is discussed manual user interface margin evaluation results.

The next tables are described margin evaluation information for all evaluated user interface elements are aligned horizontally.

Table 9- 22Margin Evolution Information when Elements are aligned horizontally

Element Name Button	***Size***		***Position***				***Margin***			
	Width	***height***	***Left***	***Top***	***Right***	***Bottom***	***Left***	***Top***	***Right***	***Bottom***
One	195	156	57	48	257	204	9	0	9	18
Two	189	174	285	48	474	222	60	0	36	18
Three	240	192	519	48	759	240	9	0	12	18
Four	225	210	801	48	1026	258	30	0	6	36

A. User Interface Elements Information

The above *table A* described user interface element margin evaluation information for all evaluated user interface elements are aligned horizontally.

Alignment	***Element Name***	***Total Width***	***Free Layout Width***	***Expected Free Space***
Horizontal one	one	849	1080-849=231	4.62
	two			
	three			
	Four			
		Total height	Free Layout height	
Vertical one	one	156	1536-156=1380	13.8
Vertical two	Two	174	1536-174=1362	13.62
Vertical three	Three	192	1536-192=1344	13.44
Vertical four	Four	210	1536-210=1326	13.26

B. Total Size, Free and Expected space Results

The above *table B* is discussed manual user interface margin evaluation results for all evaluated user interface elements are aligned horizontally.

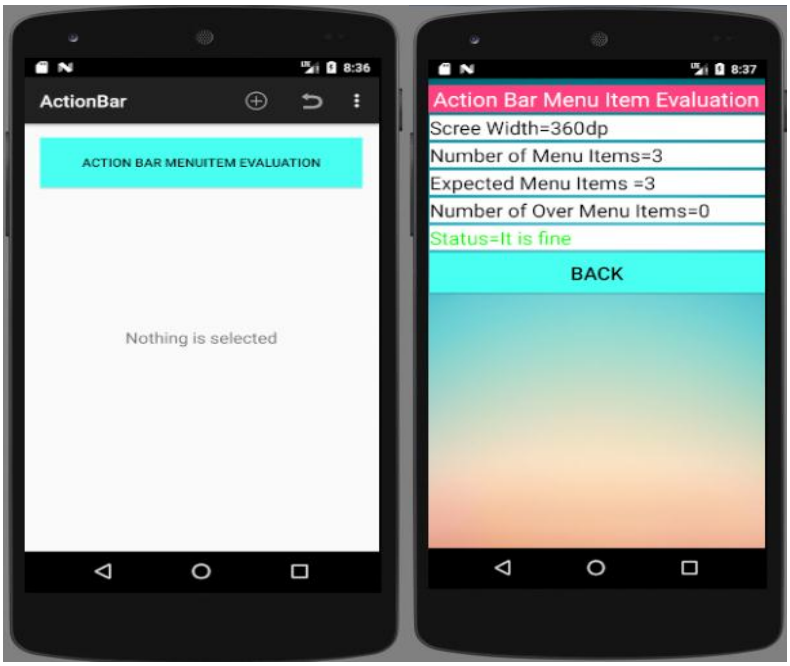


Figure 9: 6Action Bar Menu Evaluation Host Application And Evaluation Result

The above figures are described action bar menu item evaluation host application and its result. This result shows that host application has three action bar menu items and the screen width is 360 dp. According to action bar menu item evaluation guidelines, the expected and numbers of menu items are equal so the proposed framework display the status is fine message.