**Chapter one**

**Introduction**

Android operating system (OS) is widely used by smart phones and tablets. It was originally developed by a startup of the same name, Android, Inc. In 2005, as part of Google strategy to enter the mobiles domain, now it is also used in TVs and cameras. It's purchased Android and took over its development work (as well as its development team).

Google intend to make Android to be open and free, hence, most of Android ‘codes were released under the open-source Apache License, which means that anyone who wants to use Android can do so by downloading the full Android source code.

Moreover, vendors (typically hardware manufacturers) can add their own proprietary extensions to Android and customize Android to differentiate their products from others. This simple development model makes Android very attractive and has thus piqued the interest of many vendors. This has been especially true for companies affected by the phenomenon of Apple’s iPhone, a hugely successful product that revolutionized the Smartphone industry. [ Wei-MengLee 2011]

Android has taken the world by storm, everybody wants a Smartphone or tablet, and Android devices became hugely popular. Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide.

User interaction with an android device is primarily visual and tactile in nature. All these interactions took a place through the user interface of the applications that are installed on devices. Including both the built-in applications and any third-party application installed by the user. The spread of variety of Android applications, make Android developers need to make a unique interface, and a new feature in their application view, to make its special.

The version history of the Android mobile operating system began with the release of Android alpha in November 5. 2007. The first commercial version, Android 1.0, was released in September 2008. Android is continually developed by Google and the Open Handset Alliance, and it has seen a number of updates to its base operating system since the initial release.

Versions 1.0 and 1.1 were not released under specific code names, but since 2009's Android 1.5 Cupcake, Android versions have had confectionery-themed code names. Each is in alphabetical order, with the most recent major version being Android 7.0 Nougat, released in August 2016. Table 1 show the Android versions with its code name, Version number, Initial release date, API level, and API level. [ Android Security Bulletin 2017]

**Table.1 Android Versions**

|  |  |  |
| --- | --- | --- |
| **Code name** | **Version number** | **Initial release date** |
| Android (Alpha) | 1.0 | September 23, 2008 |
| Beta (known as "[Petit Four](https://en.wikipedia.org/wiki/Petit_four)" internally) | 1.1 | February 9, 2009 |
| [Cupcake](https://en.wikipedia.org/wiki/Android_Cupcake) | 1.5 | April 27, 2009 |
| [Donut](https://en.wikipedia.org/wiki/Android_Donut) | 1.6 | September 15, 2009 |
| [Eclair](https://en.wikipedia.org/wiki/Android_Eclair) | 2.0 – 2.1 | October 26, 2009 |
| [Froyo](https://en.wikipedia.org/wiki/Android_Froyo) | 2.2 – 2.2.3 | May 20, 2010 |
| [Gingerbread](https://en.wikipedia.org/wiki/Android_Gingerbread) | 2.3 – 2.3.7 | December 6, 2010 |
| [Honeycomb](https://en.wikipedia.org/wiki/Android_Honeycomb) | 3.0 – 3.2.6 | February 22, 2011 |
| [Ice Cream Sandwich](https://en.wikipedia.org/wiki/Android_Ice_Cream_Sandwich) | 4.0 – 4.0.4 | October 18, 2011 |
| [Jelly Bean](https://en.wikipedia.org/wiki/Android_Jelly_Bean) | 4.1 – 4.3.1 | July 9, 2012 |
| [KitKat](https://en.wikipedia.org/wiki/Android_KitKat) | 4.4 – 4.4.4 | October 31, 2013 |
| [Lollipop](https://en.wikipedia.org/wiki/Android_Lollipop) | 5.0 – 5.1.1 | November 12, 2014 |
| [Marshmallow](https://en.wikipedia.org/wiki/Android_Marshmallow) | 6.0 – 6.0.1 | October 5, 2015 |
| [Nougat](https://en.wikipedia.org/wiki/Android_Nougat) | 7.0 – 7.1.1 | August 22, 2016 |
| O | 8.0 | TBD |

**1.1 Android applications**

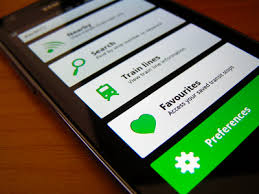
Android applications usually written by java programming language, its official language. There are around 2.7 M applications developed for android with over 50 billion downloads and this numbers are increasing. Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. For software development, Android provides Android SDK (Software Development kit). All android applications called apps. Are built on android UI (user interface) framework, app interface is the first thing a user see and interacts with. Android UI is divided into their different areas [Go Doc .Retrieved 2017]

* **Home Screen:** is the landing area when we power our phone on is highly customized.
* **All Apps** is the interface where the App instated are displayed.
* **Recent screen**: are the list of last used Apps.

**1.1.1 Android App structure and UI patterns**

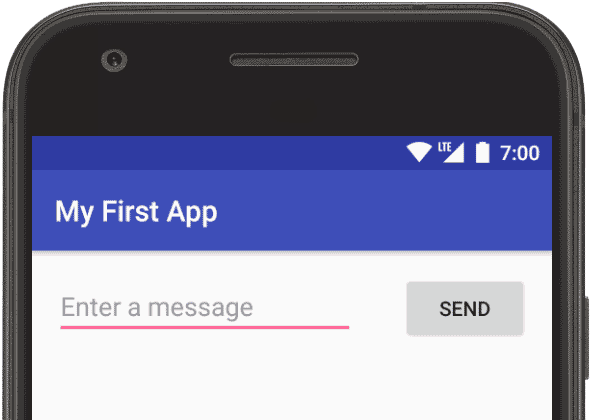
Android App are very different from each other because they are trying to address different user needs.

In general: Android App is made by Top – level view and detail level view. show in Figure 1.1.



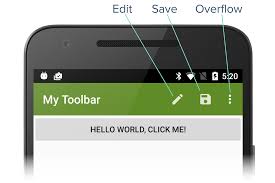
**Figure 1.1: Android App UI.**

1. **Top level view:** Is the interface that the user can’t edit or modify anything, only display information. show in Figure 1.2



**Figure 1.2: Top level view.**

1. **Detail view (low level view)** Is the interface, where the user can Interact with data directly. show in Figure 1.3



**Figure 1.3: Detail view.**

1. **Action Bar:** Is relatively new in Android, and it’s a piece of the screen. Usually on the top.

**1.1.2 Application components**

**1-Activities**

They dictate the UI and handle the user interaction to the smart phone screen.

**2-Services**

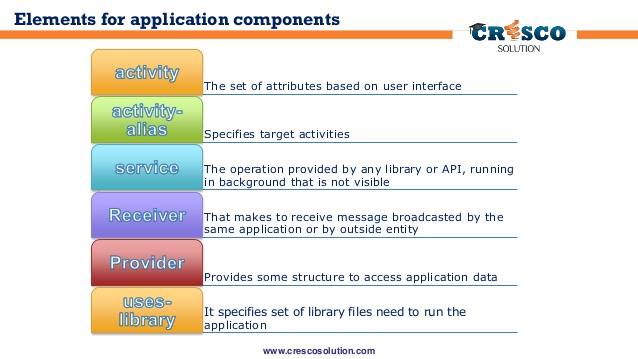
They handle background processing associated with an application.

**3-Broadcast Receivers**

They handle communication between Android OS and applications.

**4-Content Providers**

They handle data and database management issues. Figure 1.4 shows the elements for application components in android:

**Figure 1.4: android components.**

There are additional components which in android device UI will be used in the construction of above mentioned entities, their logic, and wiring between them. These components are:

**1-Fragments:** Represents a portion of user interface in an Activity.

**2-Views:** UI elements that are drawn on-screen including buttons, and lists forms

**3-Layouts:** View hierarchies that control screen format and appearance of the views.

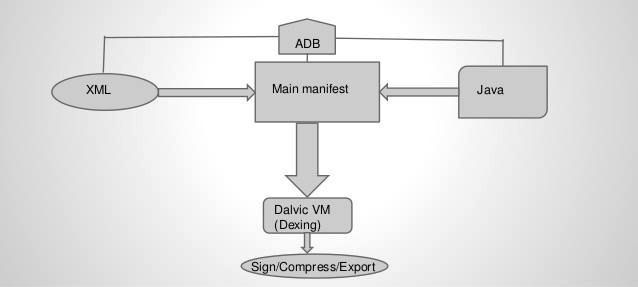
**4-Intents:** Messages wiring components together.

**5-Resources:** External elements, such as strings, constants and draw able pictures.

**6-Manifest:** Configuration files for the application. [Android Tutorial]

Android Application are composed of one or more applications components (activates, services, content providers, and broadcast receivers).

Each component performs a different role in the overall application behavior, and each one can be activated individually (even by other application). The manifest file must declare all components in the application and should also declare all applications requirements, such as the minimum version of Android required and any hardware configurations required. On-code application resources (images, strings, layout files, etc.) should include alternatives for different device configuration (such as different strings for different languages). Before the Android system can start an app component; the system must know that the component exists by reading the app's manifest file, AndroidManifest.xml. Your app must declare all its components in this file, which must be at the root of the app project directory. show in Figure 1.5



**Figure 1.5: Android application structure.**

The manifest does a number of things in addition to declaring the app's components, such as the following:

* Identifies any user permissions the app requires, such as Internet access or read-access to the user's contacts.
* Declares the minimum [API Level](https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels) required by the app, based on which APIs the app uses.
* Declares hardware and software features used or required by the app, such as a camera, Bluetooth services, or a multitouch screen.
* Declares API libraries the app needs to be linked against (other than the Android framework APIs), such as the [Google Maps library](http://code.google.com/android/add-ons/google-apis/maps-overview.html). show in Figure 1.6 present the [API Level](https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels):



**Figure 1.6:**[**API Level**](https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

**Compound Views**

Compound views (also known as Compound Components) are pre-configured View Groups based on existing views with some predefined view interaction. Compound views allow developer to add custom API to update and query the state of the compound view, for such a control, the developer define a layout file and assign it to the compound view. In the implementation of the developer, compound view the developer predefines the view interaction. Android developer would define a layout file and extend the corresponding View Group class. In This class, the developer inflates the layout file and implements the view connection logic. [Java Debugging with Eclipse Tutorial].

**Custom Views**

Custom views are typically created to provide the user interface an experience which is not possible with the default views (button, and text view, etc.), Using custom view allows the developer to do certain performance optimization, i.e., in case of a custom layout the development can optimize the layout manager for his use case.

- The android developer wants to build a new innovative and cool interaction or UI, and he cannot make it work with the current Android components: maybe the android developer need a customized view.

To make the code a little more reusable. Maybe the android developer has some logic that is cool, and he wants to use in different parts of his application, and the android developer does not want to copy-paste Code everywhere. Or maybe the android developer wants to make it more modular; has a complex UI, and want to have good software engineering, good software design, and separate things out. Custom views can help the developer with that as well.

There are also a few reasons why you might not want to use custom views. They are time-consuming, difficult and need special skills.

From the previous points, came the idea of creating website Custom View" that helps the android developers to design view custom without write code, in short time, and make it unique.

Our website is called **XML Custom View** that helps Android developers to create and custom view for their application in such an easy way and in short time, this site will generate XML code only for the view which they designed.

**XML** stands for **E**xtensible **M**arkup **L**anguage and is a text-based markup language derived from **S**tandard **G**eneralized **M**arkup **L**anguage (**SGML**).

XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data. XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML. [XML]

There are two important characteristics of XML that make it useful in a variety of systems and solutions:

**1**. XML is extensible: XML allows the developer to create his own self-descriptive tags, or language that suits to his application.

**2**. XML is a public standard: XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

**1.2 Background**

With the spread of variety of android application, android developers need to make a new interface feature in their application view. For some of the websites help them in finding the needed code and to use during designing their application.

A variety of sites that can help the developers in finding XML exists, each one has his advantage and some limitation. We will list bravely here and more detailed information will be presented in chapter 2.

**1. Android Drawable Generator:**

Android Drawable Generator is online tool to generate buttons code for Android Apps. Android API provides Drawable Resources where XML file defines geometric shapes, including colors, borders and gradients. This button is generated based on shape drawable XML code which loads faster comparing with normal PNG buttons.

Available at: http://hateum.github.io/AndroidDrawableGenerator

**2. Android Button Maker:**

Android Button Maker is online tool to generate buttons code for Android Apps. Android API provides Drawable Resources where XML file defines geometric shapes, including colors, border and gradients.

This button is generated based on shape drawable XML code which loads faster comparing with normal PNG buttons. The developer can customize button properties in setting panel and get source code.

Available at :(<http://angrytools.com/android/button/>)

**3. Drawable Designer:**

Drawable Designer is Android application helps programmers / developers to use Android in the form of designs button-shaped (rectangular - oval - circular - line) and after completing the design export to xml.

(<https://play.google.com/store/apps/details?id=com.tiraisoft.drawabledesigner&hl=en>

**4. XML Button Maker (CM Button Creator):**

The application allows the developer to quickly create buttons in your own style and save it as an XML file on the memory card for using it in his android projects.

Available at: (<https://play.google.com/store/apps/details?id=ru.articat.cmbc>)

**5. Android Shapes Generator:**

This small, site will help Android developer in save time and effort when designing interfaces for Android programs. Using visual tools to adjust the appearance and color, then it generates the xml code and can be used in our projects.

Available at :(<http://shapes.softartstudio.com/>)

**6. Dev Tools – Style Splash:**

Create the developer style for Android-application on his phone or tablet using VisX Style Splash The simplest and the most intuitive interface of the application will help the developer in creating styles faster and fan. Dynamic preview will help the developer to achieve the results he desires. There are two versions of this application:

**1- Developer Tools Style Splash**

This applications version Available at: (<https://play.google.com/store/apps/details?id=com.visx.developer_tools.appstyler_lite>).

**2- Dev Tools - Style Splash PRO**  
Available at: (<https://play.google.com/store/apps/details?id=com.visx.developer_tools.appstyler>).

**1.3 Problem Statement**

The problem handled in this project can be described as follow:

1. There exists web site that support custom view in xml but it does not support all views
2. Sites help android developers with design the button view only.
3. The developer does not use custom view because of the time consuming and require specific skills.

**1.4 Objectives of the Project**

In this project, we will develop a website that is easy to use and can help the developer that wants to create a new view to his android application in finding the needed xml code without writing it

We seek to achieve some of the objectives, these such as:

1. To design view in an android application in soft way.

2. To simplify the view design for android developers.

3. To Shorten the time for android developers.

4. To make the first, second and third points more flexible and simple.

5. The Android developers who do not have enough skills can write code XML to design view of their own with ease.

**1.5 Project Motivation**

Many problems in the process of designing view became more complex and difficult for android developers within android developer’s community, keep pace with technology, learn a new programming language Used vary widely these days, and the site that we are working on does not have a similar in my country so this made this process much flexible.

**1.6 Scope**

This project for Android developers, students, and anybody interest in android view design.

**1.7 Implementation and Modeling**

We will develop our site using HTML5, CSS, JAVASCRIPT, Bootstrap, JQUERY

ASP.Net (C#) and using java for full custom view.

That’s were on the following Browsers: Chrome, IE/Edge, Firefox, Safari, and Opera.

It is expected to have five Web pages:

**1-Main Features**

This page will display video, which explains how to use the site.

**2-Template**

If the user wants ready templates he/she can access this page and take the template of view he wants (in xml).

**3-Customize**

This page will help the developers to customize their applications.

First the user must select the element of android app interface (button or radio button) after that he/she will choose the view Appropriate to he/she want to customize his/her applications.

**4-Feedback**

The developers or the visitors for the website can send feedback about any issue that confound, by this corner in the site.

**5-Help**

For common issue in android community design view and xml code.

**1.8 Contribution**

This project (XML Custom View) contributes to

- Support of programmers and developers on the Android interfaces of their own design personally without the need of programming skills in the XML. Shorten the time programmers / developers who do not have enough time to do it themselves.

- Develop the application interfaces, to make it different, unfamiliar and more customized.

- Support most of tags used in Code XML more than previous existing works.

**1.9 Organization of the chapter**

This project consists of seven chapters organized as follows:

**Chapter.1 Introductions**

This chapter presents an introduction and background of Android app. Followed by the problem statement, the objectives, motivation, contribution and the outline of the project.

**Chapter.2 related works**

This chapter will present the theoretical background of the project and some of the similar works presented currently, advantage, disadvantage and some comparison.

**Chapter.3 system analysis**

In this chapter, a detailed analysis of the actual systems, and new system will introduced to find the new issues in the proposed system in comparison to the old systems.

**Chapter.4 system design and implementation**

This chapter contains how the proposed system is implemented and the methods used to that.

**Chapter.5 results and discussion**

This chapter will discuss the results obtained from the proposed, and implemented system. And what we have realized in this project.

**Chapter.6 testing**

In this chapter, we will introduce how we test our system from the user point of view (web site), and from the developer point of view (on android studio).

**Chapter.7** conclusions and future works.

**CHAPTER TWO**

**BACKGROUND AND RELATED WORKS**

**2.1 Introduction**

**UI – GUI**

[User Interface](https://en.wikipedia.org/wiki/User_interface) **UI** is the screen that allows the [users](https://en.wikipedia.org/wiki/User_(computing)) to devices through commands. **GUI** [Graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface) is type of [user interface](https://en.wikipedia.org/wiki/User_interface) that allows [users](https://en.wikipedia.org/wiki/User_(computing)) to interact through graphical [icons](https://en.wikipedia.org/wiki/Computer_icon) and visual Elements [MSCOM 2007] as in Figure 2.1.

|  |  |
| --- | --- |
| **commandline** | **gui** |

**Figure 2.1: the different between CLUI and GUI**

**Android UI**

A mobile OS, like [Android](https://techterms.com/definition/ios) , are designed to allow the user to interact with the device’s [apps](http://whatis.techtarget.com/definition/mobile-app), features, content and functions by [touchscreen](https://techterms.com/definition/touchscreen) and supports touch commands like [swiping](https://techterms.com/definition/swipe) and pinching to zoom in or zoom out.

**Android UI Components**

The Android provides a variety of pre-built **UI** components such as structured layout objects and **UI** controls that allows the developer to build the graphical user interface for his app.

All **UI** elements in an Android app are built using [**View**](https://developer.android.com/reference/android/view/View.html) and [**ViewGroup**](https://developer.android.com/reference/android/view/ViewGroup.html) objects. **A** [**View**](https://developer.android.com/reference/android/view/View.html) **is an object that draws something on the screen**.

A [**ViewGroup**](https://developer.android.com/reference/android/view/ViewGroup.html) is an object that holds other [**View**](https://developer.android.com/reference/android/view/View.html) and [**ViewGroup**](https://developer.android.com/reference/android/view/ViewGroup.html) objects in order to define the layout of the interface.

We can declare a UI Components in two ways:

* **Declare UI elements in XML:** Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses.
* **Instantiate layout elements at runtime:** The application can create View and **ViewGroup** objects (and manipulate their properties) programmatically by java Code.

The advantage to declaring the UI in XML is that it enables the developer to better separate the presentation of the application from the code that controls its behavior.

The UI descriptions are external to the application code, which means that we can modify or adapt it without having to modify the source code and recompile. For example, we can create XML layouts for different screen orientations, different device screen sizes, and different languages. Additionally, declaring the layout in XML makes it easier to visualize the structure of the UI, so it's easier to debug problems.

In this project, we will focus on this part, where our web site will help the developer to declare his own layout in XML without the need of modifying the source code of the application.

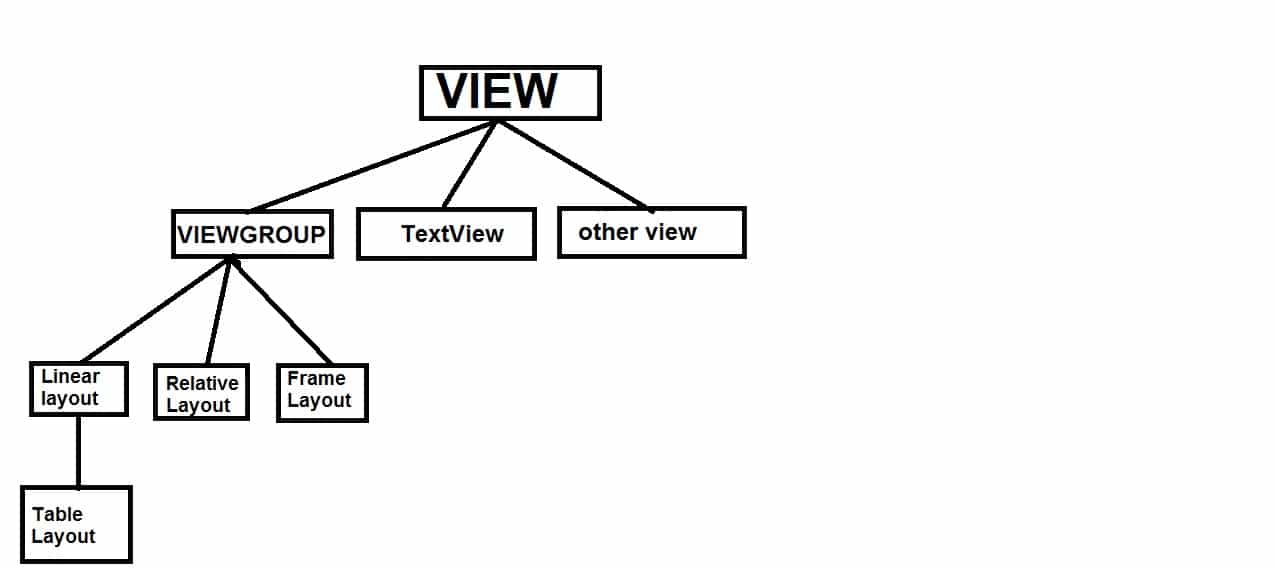
**Android UI Layout**

The basic building block for user interface is a **View** object which is created from the View class and occupies a rectangular area on the screen and is responsible for drawing and event handling.

**View** is the base class for widgets, which are used to create interactive **UI** components like buttons, text view, etc. (first level).

The **ViewGroup** is a subclass of **View** and provides invisible container that hold other Views or other **ViewGroups** and define their layout properties. (Second level).

At third level, we have different layouts which are subclasses of **ViewGroup** class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using **View/ViewGroup** objects in code or can be declared using simple XML file. Figure 2.2.

****

**Figure 2.2: the hierarchical view**

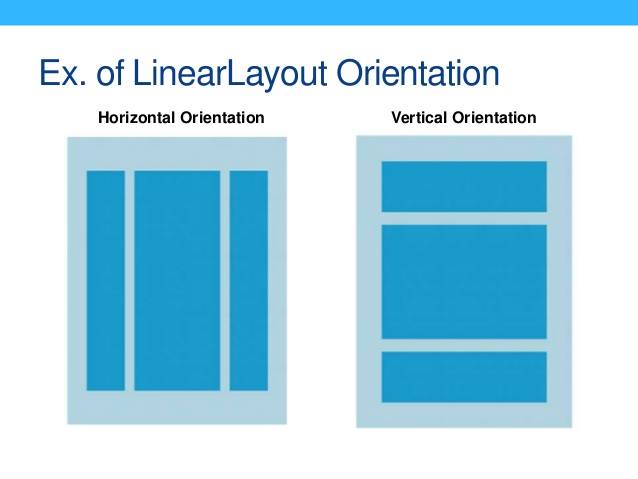
The easiest and most effective way to define the layout is in **XML** file, **XML** offers a human-readable structure for the layout.

**Android Layout Types**

There are number of Layouts provided by Android which are used in all the Android applications to provide different view, look and feel [Android Layout Types].

* [**Linear Layout**](https://www.tutorialspoint.com/android/android_linear_layout.htm)

Linear Layout is a view group that aligns all children in a single direction, vertically or horizontally. Figure 2.3

****

**Figure 2.3: linear layout**

In **vertical linear layout,** all child views are arranged in vertical list. So, a vertical list will only have one child per row, no matter how wide they are.

In **horizontal linear layout**, all child views are arranged in one row horizontally and a horizontal list will only be one row high (the height of the tallest child, plus padding).

* [**Relative Layout**](https://www.tutorialspoint.com/android/android_relative_layout.htm)

Relative Layout is a view group that displays child views in relative positions. Figure 2.4

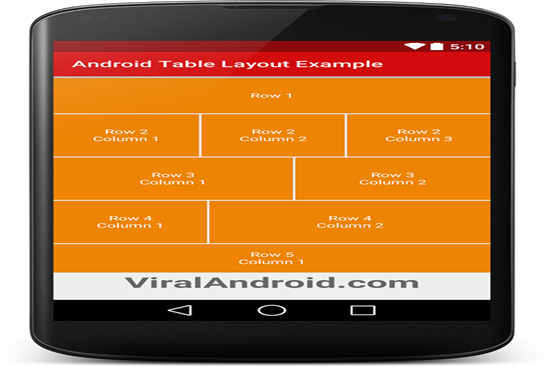
|  |
| --- |
| 2 |

Relative layout child control placement is defined using rules. These rules define how the controls within the relative layout are displayed. For the complete list of rules for relative layouts, it means that child controls, such as ImageView, TextView, and Button controls, can be placed above, below, to the left or right, of one another. Child controls can also be placed in relation to the parent (the relative layout container), including placement of controls aligned to the top, bottom, left or right edges of the layout.

* [**Table Layout**](https://www.tutorialspoint.com/android/android_table_layout.htm)

Table Layout is a view that organize groups views into rows and columns. Figure 2.5

Table Layout is a ViewGroup that displays different child views like TextView, ImageView, EditText, Button, etc. in rows and columns.



**Figure 2.5: Table Layout**

* [**Absolute Layout**](https://www.tutorialspoint.com/android/android_absolute_layout.htm)

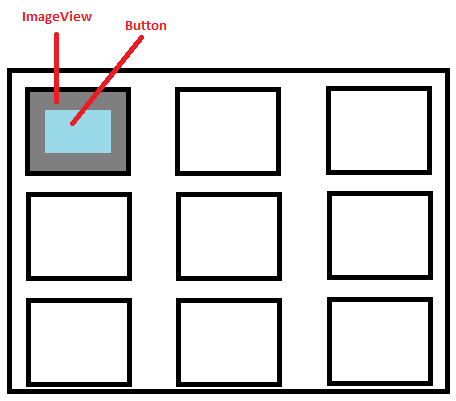
Absolute Layout enables the developer to specify the exact location of its child, in absolute layout, we can specify the exact coordinates of each control that we want to place.Figure 2.6



**Figure 2.6: Absolute Layout**

* [**Frame Layout**](https://www.tutorialspoint.com/android/android_frame_layout.htm)

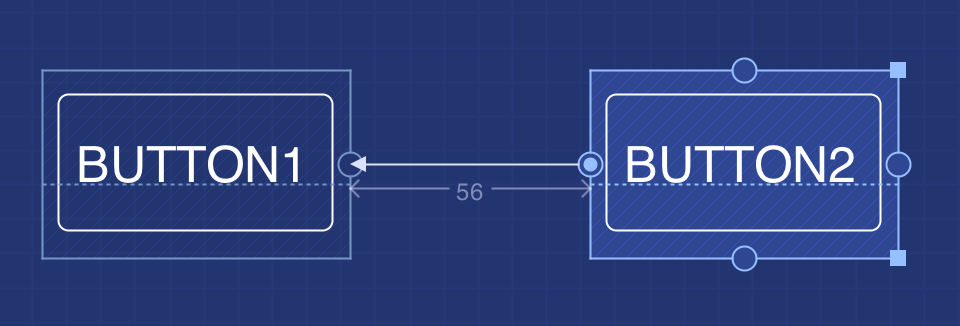
The Frame Layout is a placeholder on screen that can be used to display a single view, they are designed to block an area on the screen. Frame Layout should be used to hold child view, because it can be difficult to display single views at a specific area on the screen without overlapping each other.Figure 2.7



**Figure 2.7: Frame Layout**

* **Constraint Layout (CL)**

Constraint Layout is a layout that will feel very similar to Relative Layout at first glance, but CL is more powerful and versatile .CL aims to make developing layouts easier, faster, and more intuitive, while improving performance at runtime.Figure 2.8



**Figure 2.8: Constraint Layout**

**2.2 Theoretical background**

Android UI Controls, Input controls are the interactive components in the app's user interface, Android provides a wide variety of controls can be used in the UI, such as button, text field, check box, zoom button, toggle button, and many more.

Each input control supports a specific set of input events so the developer can handle events such as when the user enters text or click a button [Android UI Controls].

1. [**Button**](https://developer.android.com/guide/topics/ui/controls/button.html): A push-button that can be pressed, or clicked, by the user to perform an action. Figure 2.9



**Figure 2.9: Button**

1. [**TextView**](https://developer.android.com/reference/android/widget/TextView.html): Displays text to the user and optionally allows them to edit textview. **Figure 2.10**

 **Figure 2.10: TextView**

1. **Edit Text**: EditText is a thin veneer over TextView that configures itself to editable. Figure 2.11

****

**Figure 2.11: Edit Text**

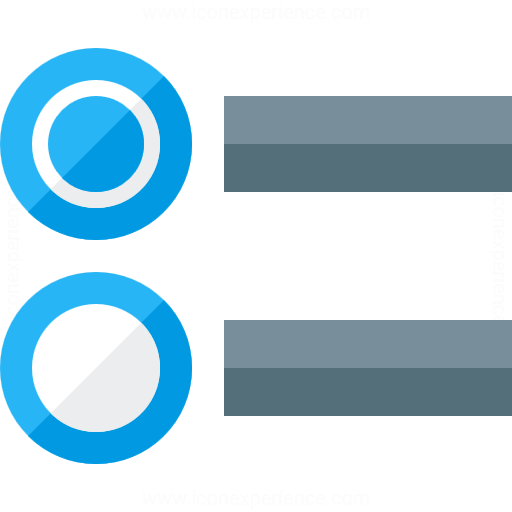
**Note:** An EditText is a generic input widget, whereas an auto Complete Textview uses prediction to suggest input alternatives. Connect it to an array or a cursor and it will filter the content based on what the user starts to type.

1. [**Check Box**](https://developer.android.com/reference/android/widget/CheckBox.html)**:** A checkbox is a specific type of two-state button that can be either checked or unchecked, is used where the user can choose more than one choices. Figure 2.12

****

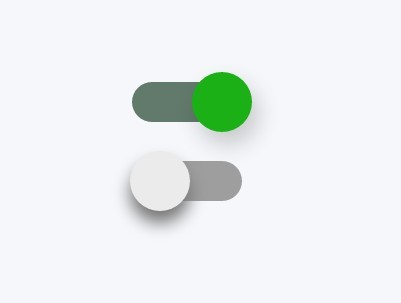
**Figure 2.12: Check Box**

1. [**Radio button**](https://developer.android.com/guide/topics/ui/controls/radiobutton.html) **:**Similar to checkboxes, except that only one option can be selected in the group, are always couple, if one is checked (true) the other cannot be checked (false). Figure 2.13

****

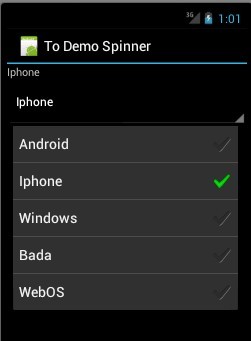
**Figure 2.13: Radio button**

1. [**Toggle Button**](https://developer.android.com/guide/topics/ui/controls/togglebutton.html)**:** An on/off button with a light indicator, is used to activate or deactivate an application or a siting.Figure 2.14

****

**Figure 2.14: Toggle Button**

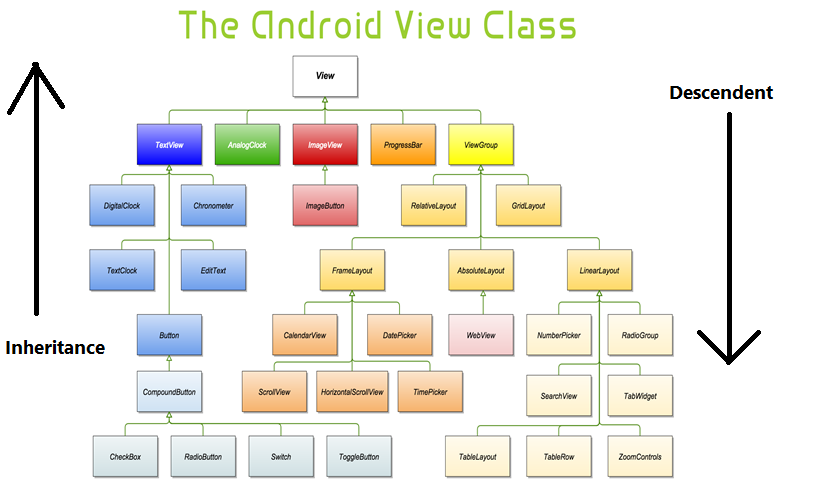
1. [**Spinner**](https://developer.android.com/guide/topics/ui/controls/spinner.html)**:** A drop-down list that allows users to select only one value from a set.Figure 2.15

****

**Figure 2.15: Spinner**

**2.3 Attributes UI Components**

Every **View** and **ViewGroup** object supports their own variety of XML attributes, some attributes are specific to a View object (for example, **TextView** supports the text Size attribute), but these attributes are also inherited by any View objects that may extend this class. Some are common to all View objects, because they are inherited from the root View class (like the id attribute). And, other attributes are considered "layout parameters," which are attributes that describe certain layout orientations of the View object, as defined by that object's parent **ViewGroup** object.Figure 2.16

****

**Figure 2.16: Android Inherited Components**

**Android Attribute view**

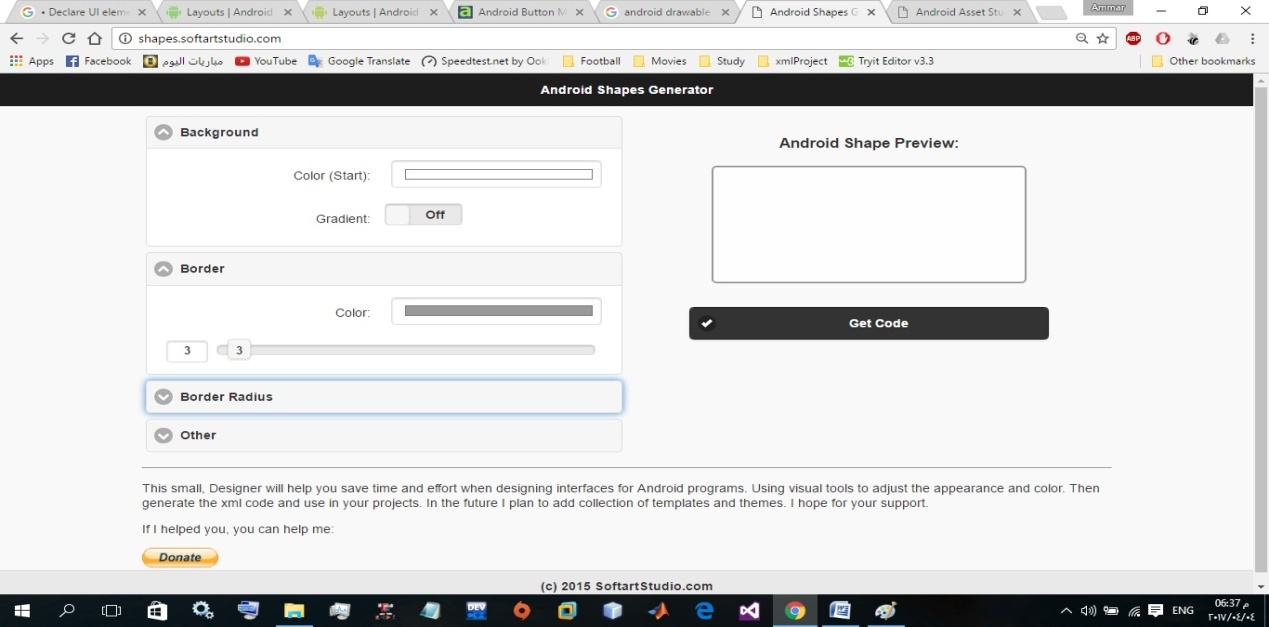
* **Common Attributes**
* **Android: clickable** attribute is an attribute two definition of that view reacts to click events, this attribute is a used to button, radio button and not to text view.
* **Android: onClick** is an attribute that is name of the method in this View's context to invoke when the view is clicked, this attribute is a used to button, radio button and not text view.
* **Inherited Attributes**
* Android: Alpha property of the view, as a value between 0 (completely transparent) and 1 (completely opaque), this attribute is a used to TextView.
* **Specific Attributes**
* **Android: TextOn** Text to use when the switch is in the checked/"on" state, this attribute is an only used Switch.

**2.4 Related Application**

In searching about new ideas for our project we have found some related works to our project but not equal. We will list some of them.

1. **Android Shapes Generator (website):**

This is a small web site that helps Android developers in save time and effort when designing interfaces for Android programs. Using visual tools to adjust the Appearance and color. Then, it generates the xml code and can be used in projects. This site has no popularity between the android developers because it doesn't support a lot of attribute (few attribute) like the change the shape and more of attribute that doesn't supported. The site has 41.80% visitor, 2248970 global traffic, 335494 India traffic, and Bounce Rate is 72.70% on Alexa Website Traffic Statistics. Available at: (<http://shapes.softartstudio.com/>)Figure: 2.17.



**Figure 2.17: Android Shapes Generator**

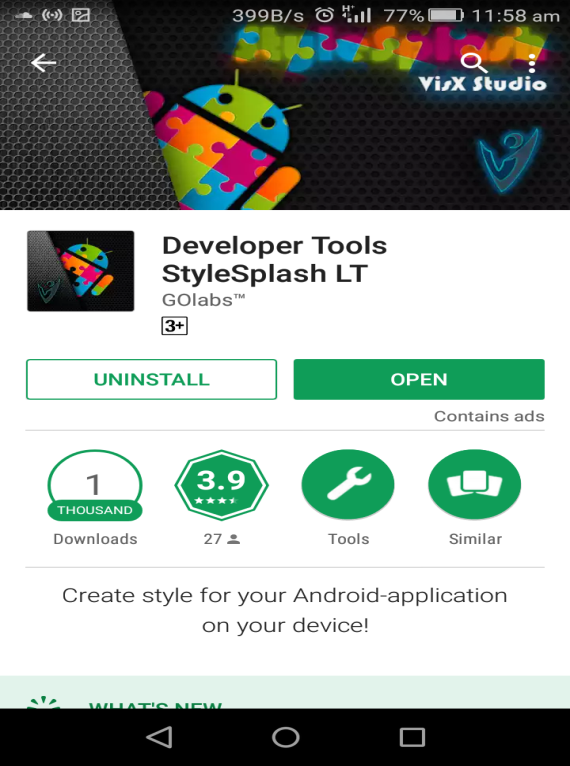
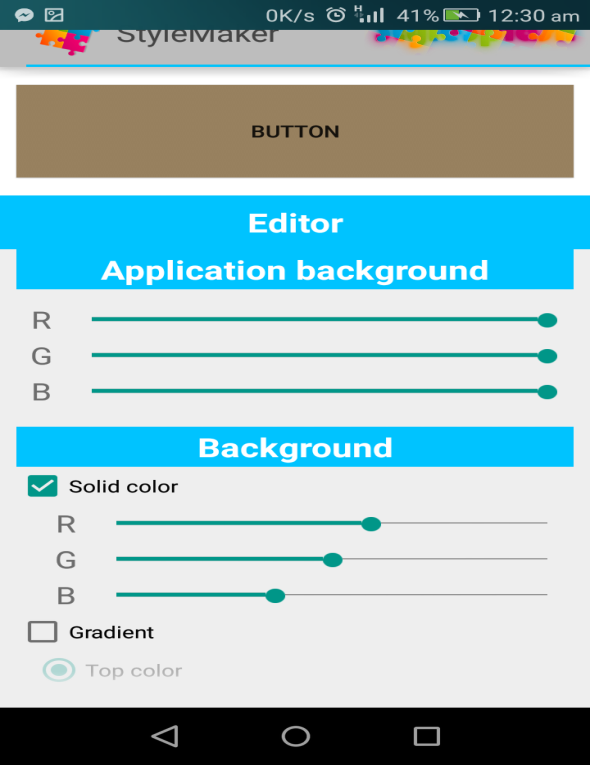
1. **Dev Tools – Style Splash (app):**

Create our own style for Android-application on our phone or tablet using VisX Style Splash, the most simple and intuitive interface of the application will help in creating styles fast and fan, Dynamic preview will help in achieve the results we desire. There are two versions of this application: -

* **Developer Tools Style Splash LT**

This is a trial version of the application, cannot save. Xml files in this version. App features are, it is free, support multi-language interface (Russian and English), Creating styles, create gradients Select a color using the RGB palette and save it to the Library, and it is Intuitive interface, Have Color library will allow the developer to keep those colors that still useful for him, installed (1000 – 2000) and the Last updated was 14/ 9 / 2014, Current Version 1.0.9-082014-lite, Requires Android2.3.3 and up, well-known developer. Figure 2.18

Available at: (https://play.google.com/store/apps/details?id=com.visx.developer\_tools.appstyler\_lite)



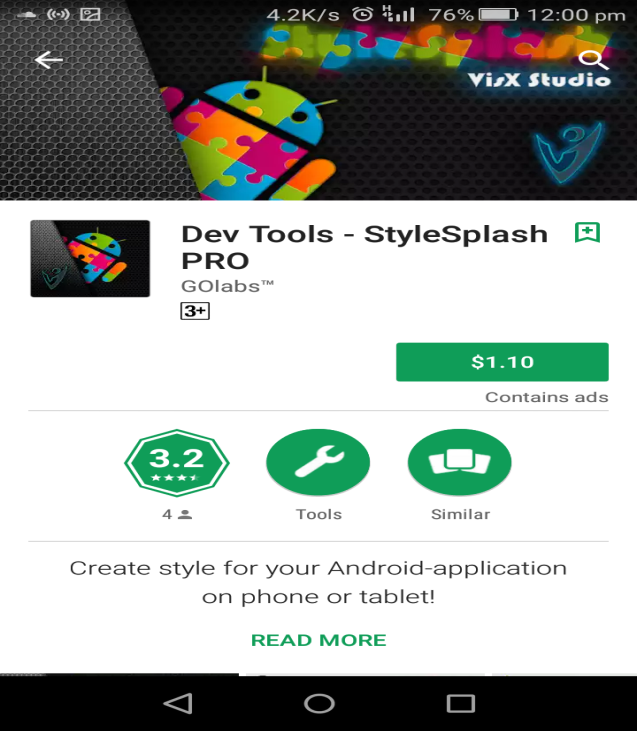
**Figure 2.18: Dev Style Splash LT**

* **Dev Tools - Style Splash PRO:**

Features of App Is not free App & its price is (1.11 $) support multi-language interface Russian and English.

Creating and saving styles in .xml files, and Create and maintain gradient in .xml files, select a color using the RGB colors palette and save it to the Library, Intuitive interface, saved styles adapted to use in your projects (just copy the style from your device to the "drawable" folder of your project and assign it to the widget), Color library will allow you to keep those colors that you still will be useful, Last updated 14/9/2014, Current Version1.0.9-082014-pro, Requires Android 2.3 and up, well-known developer. Figure 2.19

Available at: (https://play.google.com/store/apps/details?id=com.visx.developer\_tools.appstyler)



**Figure 2.19: Dev Style Splash pro.**

1. **Android Drawable Generator (website):**

Android Drawable Generator is online tool to generate buttons code for Android Apps, Android API provides drawable Resources where XML file defines geometric shape, colors, border and gradients. This button is generating based on shape drawable XML code which load faster compare to normal PNG buttons. This site has some of issue:

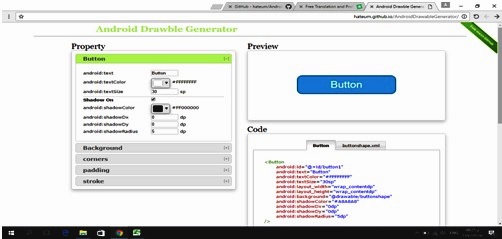
Different between some of initial value for the element that it used for custom view and the effect on the Button that display front the developer (that make the developer be confused because the conflict between the code and Button that display front him) like: Color Stroke, Background Color, and Shadow radius.

Also, there is some of attribute didn't work or some of it work in a wrong way like: End color, Center Color, Start Color, Shadow DY, Shadow DX, and Width Strake.

When the developer changes the Background Color value to Solid, that wouldn't make any change on the code, so the Background Color be always gradient.

All previous problem make the Android developer be confused when he designing the view, and cost him a lot of trouble when move the code to the work environment like: Android Studio or Xamarin.

Available at: (<http://hateum.github.io/AndroidDrawableGenerator>). Figure 2.20

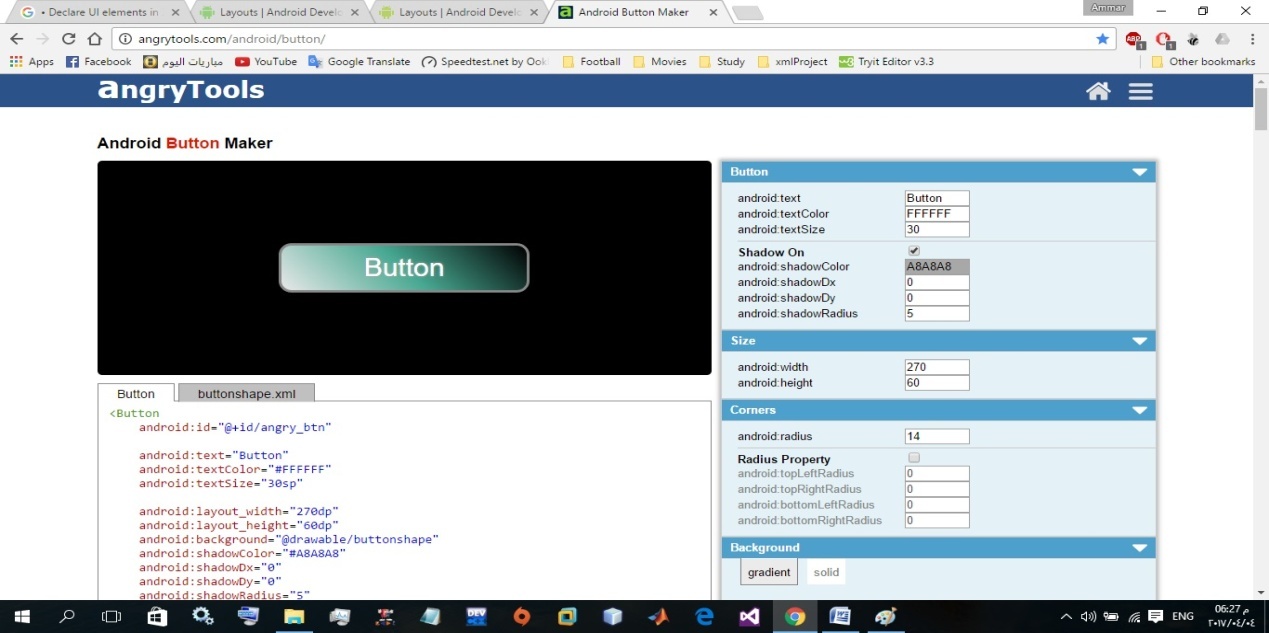


**Figure 2.20: Android Drawable Generator**

1. **Android Button Maker Angry Tools (website):**

Android Button Maker is online tool to generate buttons code for Android Apps. Android API provide Drawable Resources where XML file defines geometric shape, including colors, border and gradients. These button is generating based on shape drawable XML code which load faster compare to normal PNG buttons. You can customize button properties in setting panel and get source code. The site has 30.30% visitors, 147,907global rank, and 33007 India rank on Alexa website traffic Statistics.

Available at: (<http://angrytools.com/android/button/>)Figure: 2.21



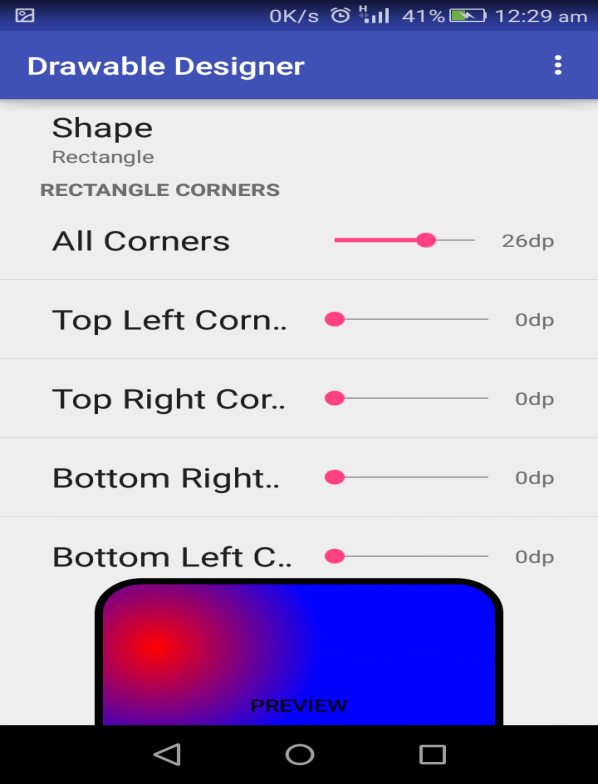
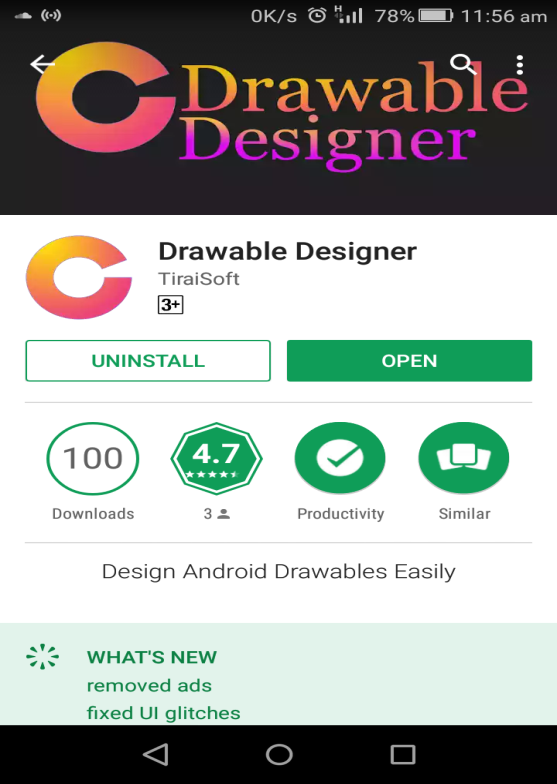
**Figure 2.21: Android Button Maker.**

1. **Drawable Designer (app):**

Features Is an android application, allows to design a different button shape like(rectangle, oval, line, ring).in addition, it is possible to modify some of button aspects like (Rectangle shape corner radius ring shape, radius and thickness)ring radius can be in ratio or dip unit, Solid color, Gradient (linear, radial, sweep, Stroke (solid, dash), Padding, Shape size, Shape level and gradient level preview, Live preview, Export to xml it is free, and support only button design and is does not contain a view, this app has a problem in padding attribute that doesn't work, like padding top, right, left, and bottom. this application requires android 4.1.

The current version is 1.2.9 and the last update 24/8/2016 and is installs 100-200 devices. Figure 2.22

Available at: (<https://play.google.com/store/apps/details?id=com.tiraisoft.drawabledesigner&hl=en>)

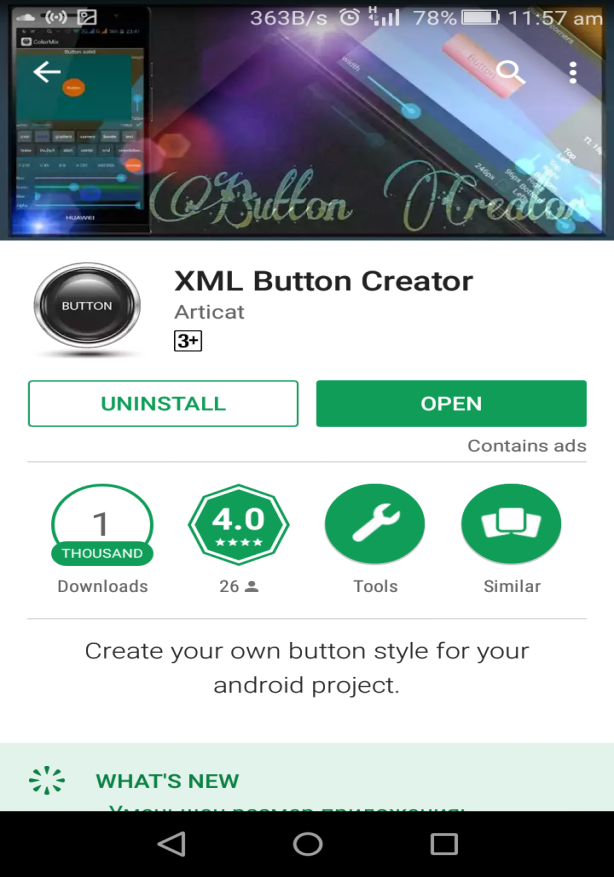
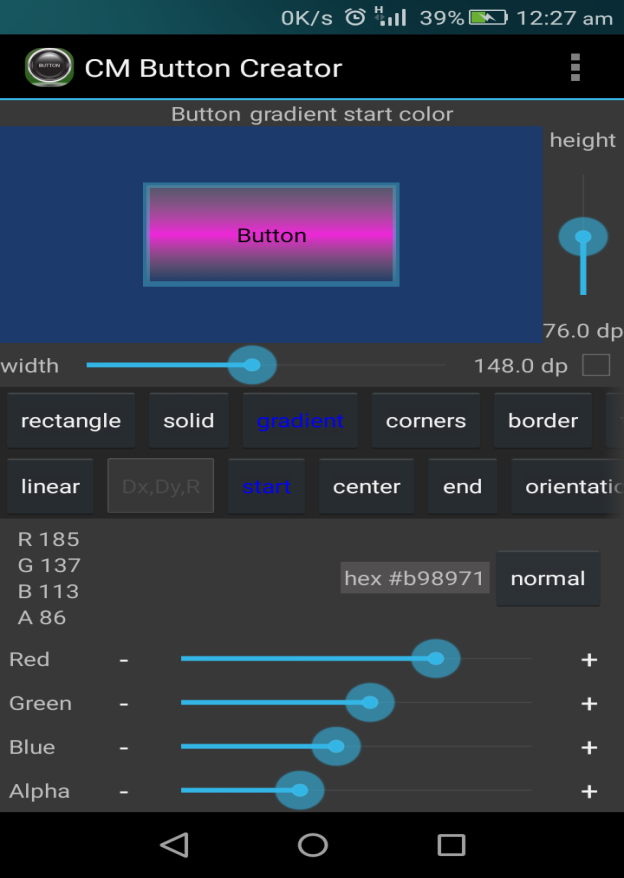


**Figure 2.22: Drawable Designer.**

1. **XML Button Maker CM Button Creator (app):**

This application allows the developer to quickly create buttons in his own style and save it as an XML file on the memory card for use in his android projects, but this application has bad interface (between the app and the user) that not usable. Some features of the XML Button Maker:

Free Application, has a single interface to design the only button, it does not contain a View and Requires Android 2.2, well-known developer, the current version is 1.2, and Last updated 10/2/2015, downloads (1000-2000). Available at: <https://play.google.com/store/apps/details?id=ru.articat.cmbc> .**Figure 2.23**

****

**Figure 2.23: XML Button Maker (CM Button Creator)**

Table 2.1. Shows the differences between all related works apps & website.

**Table 2.1: differences between related works**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **XML Custom View** | **Dev Toolsstyle splash** | **Android Shape Generator** | **XML Button Maker CM** | **Drawable Designer** | **Android Button Maker** | **Android Drawable Generator** | **#** |
| Supports All tags that is interest in view design and support new shape. | Supports Multi-language interface  And support design for normal state, pressed state and focused state |  | Support design for normal state and pressed state |  |  |  | **Advantage** |
|  | Some release of this app is Not Free (paid)  Need Version Pro for generate Code | Small Application,  Supports very limited number of attribute. | it's not usable,  and weak in Arrangement elements of interface | Some attributes not Work. Example padding left, Right, top, and bottom. |  | Exist problems in app make the developer be confused because the conflict between the code and Button that display | **Disadvantage** |

**Table 2.2 Shows the displays all attributes used and not used by related works**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tag** | **Attribute** | **Android Drawable Generator** | **Android Button Maker** | **Drawable Designer** | **XML Button Maker CM** | **Android Shape Generator** | **Dev Toolsstyle splash** |
| Size | width | × | √ | √ | √ | × | × |
| height | × | √ | √ | √ | × | × |
| Corners | radius | √ | √ | √ | × | × | √ |
| bottom left radius | √ | √ | √ | √ | × | √ |
| bottom right radius | √ | √ | √ | √ | × | √ |
| top left radius | √ | √ | √ | √ | × | √ |
| top right radius | √ | √ | √ | √ | × | √ |
| Stroke | width | √ | √ | √ | √ | × | √ |
| color | √ | √ | √ | √ | × | √ |
| dash Gap | × | × | √ | √ | × | × |
| dash width | × | × | √ | √ | × | × |
| Solid | color | √ | √ | √ | √ | × | √ |
| Gradient | type | √ | × | × | × | × | × |
| star color | √ | √ | √ | √ | √ | √ |
| center color | √ | √ | √ | √ | × | √ |
| end color | √ | √ | √ | √ | √ | √ |
| center x | √ | √ | √ | √ | √ | √ |
| center y | × | × | √ | √ | √ | √ |
| linear angle | √ | √ | √ | √ | × | × |
| radial radius | × | × | × | × | × | × |
| Padding | top | √ | √ | √ | × | √ | √ |
| bottom | √ | √ | √ | × | √ | √ |
| left | √ | √ | √ | × | √ | √ |
| right | √ | √ | √ | × | √ | √ |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tag** | **Attribute** | **Android Drawable Generator** | **Android Button Maker** | **Drawable Designer** | **XML Button Maker CM** | **Android Shape Generator** | **Dev Toolsstyle splash** |
| Button | id | × | × | × | × | × | *×* |
| text | √ | √ | × | × | × | *×* |
| text color | √ | √ | × | √ | × | *√* |
| text size | √ | √ | × | √ | × | *√* |
| text style bold | × | × | × | × | × | *×* |
| text style italic | × | × | × | × | × | *×* |
| gravity | × | × | × | × | × | *×* |
| font family | × | × | × | × | × | *×* |
| layout-width | × | √ | √ | √ | × | *√* |
| layout-height | × | √ | √ | √ | × | *√* |
| enable | × | × | × | × | × | × |
| alpha | × | × | × | × | × | × |
| background | √ | √ | √ | √ | × | √ |
| text all caps | × | × | × | × | × | × |
| radius shadow | √ | × | × | × | × | × |
| delta x shadow | √ | × | × | × | × | × |
| delta y shadow | √ | × | × | × | × | × |
| color shadow | √ | × | × | × | √ | × |
| Shape | rectangle | rectangle | rectangle, oval, line, ring | Rectangle, oval | rectangle | rectangle |

**Table 2.3: attributes in related works**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tag** | **Attribute** | **Android Drawable Generator** | **Android Button Maker** | **Drawable Designer** | **XML Button Maker CM** | **Android Shape Generator** | **Dev Tools style splash** |
| Text View | id | × | × | × | × | × | × |
| text | × | × | × | × | × | × |
| text color | × | × | × | × | × | √ |
| text size | × | × | × | × | × | √ |
| text style bold | × | × | × | × | × | × |
| text style italic | × | × | × | × | × | × |
| gravity | × | × | × | × | × | × |
| font family | × | × | × | × | × | × |
| layout-width | × | × | × | × | × | √ |
| layout-height | × | × | × | × | × | √ |
| background | × | × | × | × | × | √ |
| lines | × | × | × | × | × | × |
| Text all caps | × | × | × | × | × | × |
| radius shadow | × | × | × | × | × | √ |
| delta x shadow | × | × | × | × | × | √ |
| delta y shadow | × | × | × | × | × | √ |
| color shadow | × | × | × | × | × | √ |

**2.5 Recognition of needs**

From the previous tables, we can conclude that must of button attribute and text view are not supported by the current web site or application. That demonstrates the need of a web site or application that support the (missed) attribute and that add new attribute and new features not studied in the previous works. After get custom code and view code that would need more editing on view attribute in Work environments as where been in previous works that focus on attribute which is specialist in custom view.

The investigation about the presented site and what they offer were the main goal behind our work that is a result of our need as android developers to find something ready to use without spending a lot of time in trying and writing new codes for unique features.

All previous works support button view only, except Style Splash support TextView. And no one of this works support save the action for the android developers that we will support in our project, and we will provide to the android developers design the Button view, EditText view, TextView, and Radio Button view designing and costuming for their applications view.

**CHAPTER THREE**

**PROJECT METHODOLDGY**

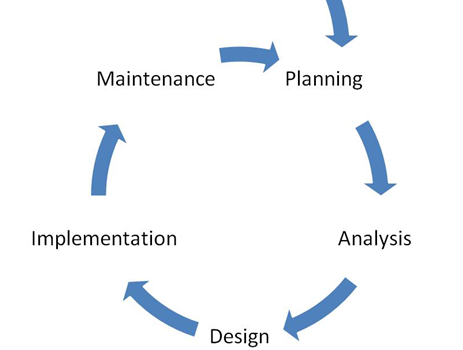
Information technology can mean the different between success and failure, can be a combination of hardware and software and services used to communication or to share information.

The world is moving towards diversity and difference, our vision in this project is to achieve the diversity in the idea of the project where our idea is slightly different from typical graduation projects, and the different in the goal that is to help the android developer in creating or developing a different view for his application that differs from the usual.

This chapter provides the steps we have taken to design our web site. Information system is a combination between system and information, system is a set of related components that produces specific results. When the user interacts with this component is defined the functional requirements of the system, and when the system shows its properties is defined the nonfunctional requirements of the system.

Information means the data how is managed and handles and processed in the system that is represented in DFD (Data Flow Diagram) and ERD (Entity Relation Diagram). And so on.

This chapter and the following two chapters will provide our system life cycle that starts from planning, analyzing, design, implementation, and finally testing show in (figure 3.1).



**Figure 3.1: System Life Cycle**

**3.1 Project Planning**

A system development methodology refers to framework that is used to structure, plan and contend the process of developing a system. It consists of the phases and actions taken to make the system come to light; this plan represents **Agile system** developments. Agile system development is an iterative software development which refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. Agile methods or agile processes generally promote a disciplined project management process that encourages frequent inspection and adaptation, leadership philosophy that encourages teamwork, self-organizing and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality software. Figure 3.2



**Figure 3.2: Agile model**

Figure 3.2, represent the agile model that is composed on different phases. In this model phases are executed sequentially and in backtracking mode that means the system all the time performs a testing of the previous phases.

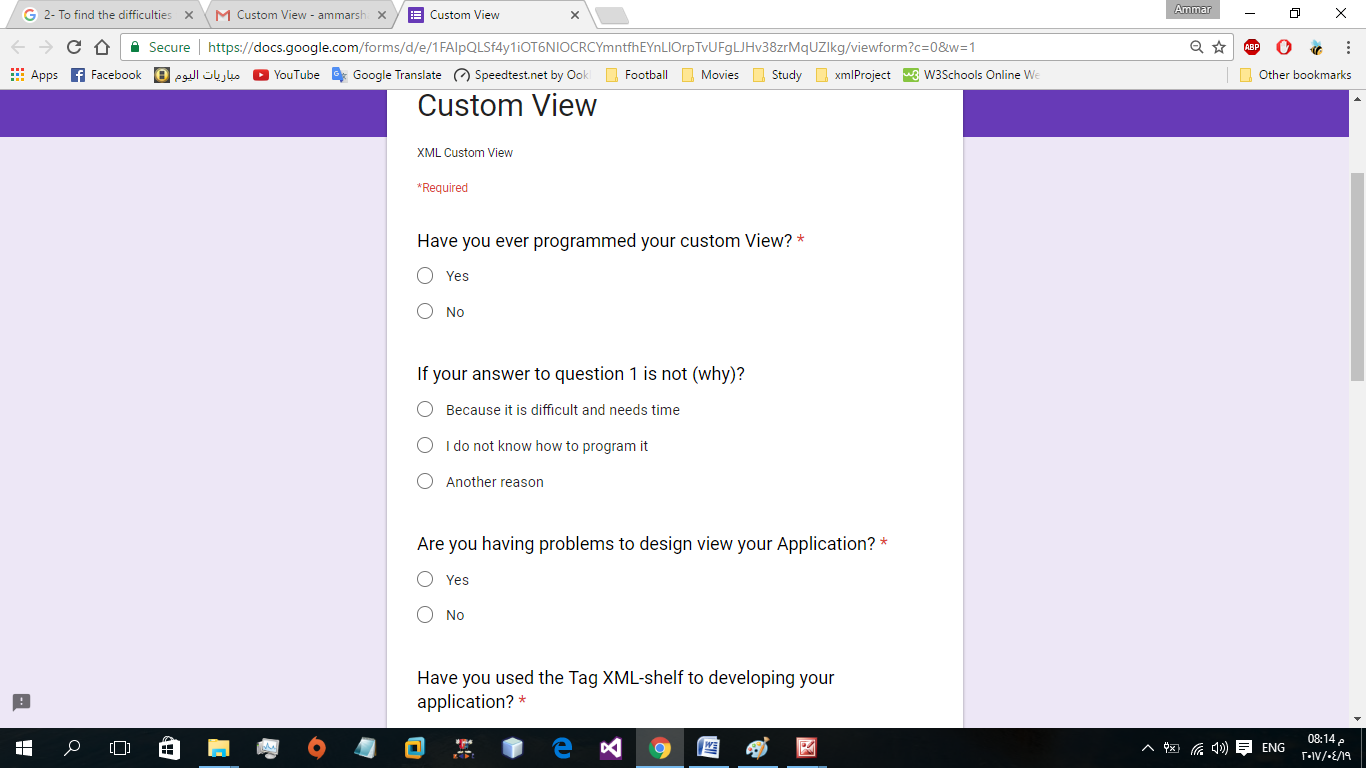
**Requirements phases:**

This is the initial phase of the development process where in the development team works closely with the customer to determine the customer's requirements for the product. The requirements phase identifies the functionality, performance levels, and other characteristics which the product must satisfy in order for it to be acceptable to the customer. The requirements developed in this phase serve as a foundation for the remaining phases of the development process, and as the customer acceptance criteria.

**3.1.1 Feasibility Study**

A feasibility study is the total of the actions can be taken and the questions can be asked to determine whether an idea, thought or plan is likely to succeed. So, we are make a questionnaire that treats the following criteria:

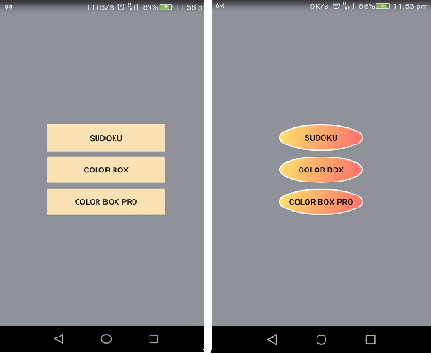
1. Find out the opinions of the android developers/programmers in customizing the android application view. Figure 3.3
2. To find the difficulties faced by the programmers in designing his own view application.

****

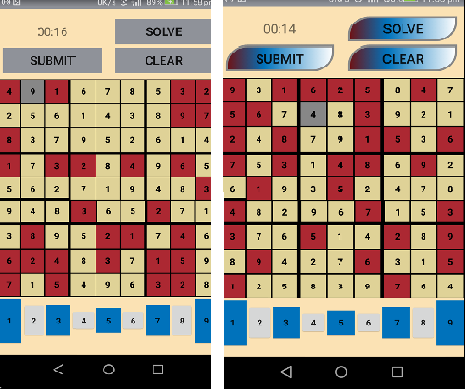
**Figure 3.2: part of the questionnaire**

After presenting the questionnaire to a group of developers / programmer’s android it has been concluded that, most Android developers / programmers want to use Custom View but, are not able to do that because of the need of great time and high skills in XML programming. And they need sites that support Custom View to reduce the time and efforts on Android developers, and to provide their unique and personal application needs.

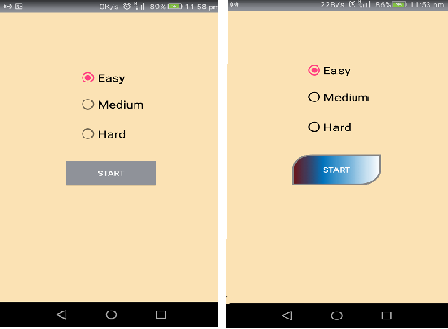
To study utility of our system we designed two android applications (Figure3.4 (a,b,c)), on left without using Custom View, and on the right with using Custom View.



**Figure 3.4.A: button design**

****

**Figure 3.4.B:**

****

**Figure 3.4.C:**

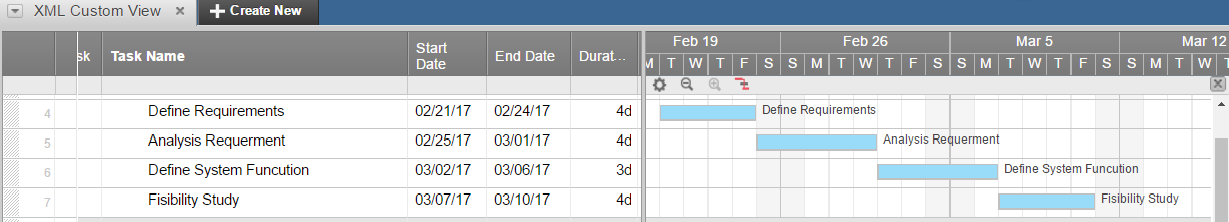
As mentioned in chapter 2, not all the actual applications support customization, and if it supports, it supports only for button. Figure 3.4.b, on the right show how look like an application with different button view, it is clear the different in the button view. In our project we will support Button, Textview, Edittext, Radio button, and Shape view.

After comparing the two applications, we noticed that application with Custom View gave the application a special form. We think that Custom View is very important system because of the increasing number of applications at this time so this site will solve the problem of similarity in applications design. Many of the Android developers like to design a unique view for their application but they cannot to luck in possibilities or due to weakness in their programming skills.

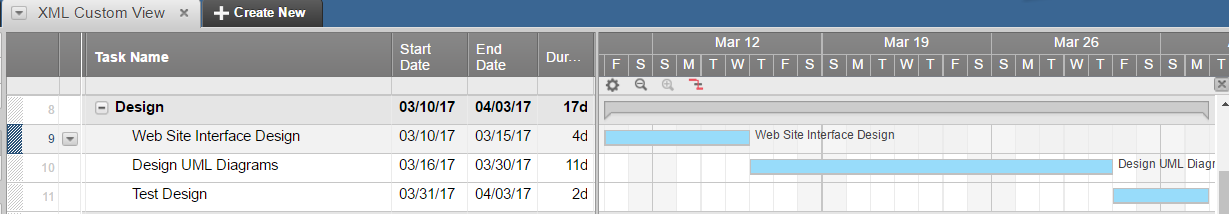
**3.1.2 Gantt Chart**

Gantt chart are useful in planning how long a project should take time and helping to organize the sequence of events by laying them out in the order in which the tasks need to be completed.

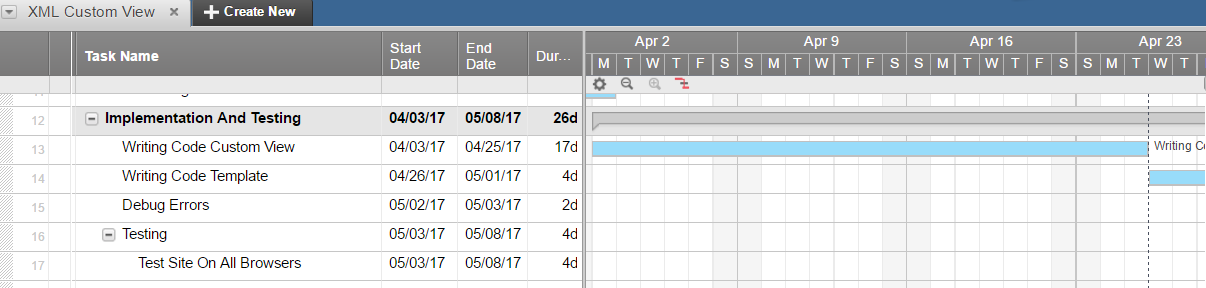
We built our Gantt chart step by step. That means, in each week we register what we have done and how many time it takes and we organize it in the following chart.

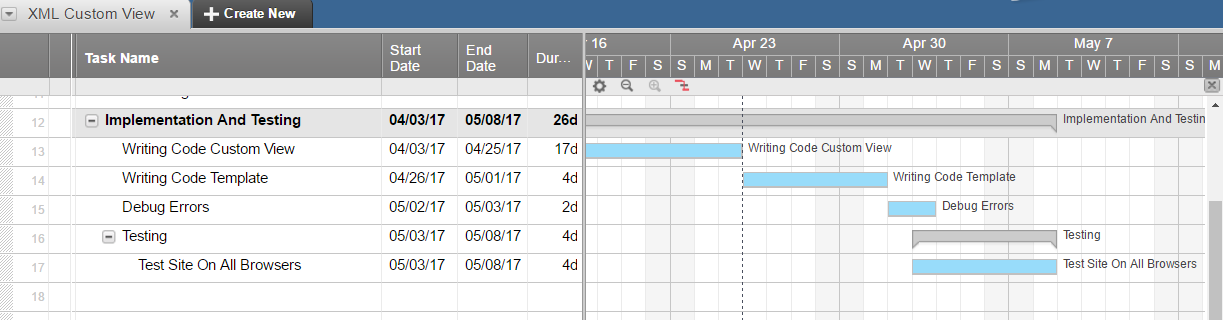


**Figure 3.5: Planning**



**Figure 3.6: Design**



**(a) **

**(b)**

**Figure 3.7. (a,b): Implementation & Testing**

**3.2 Analysis of the New System**

The analysis phase defines the requirements of the system, independent of how these requirements will be accomplished.

**3.2.1 User Requirements**

**- Functional Requirements**

Means how the system behaves from user point of view. Consider that our system is informative system, so it performs the following functions:

1. The first is to display information to the user
2. The second is to allow the user to choose what they want. (customize page)

So, the system from the user point of view is informative and selective, informative when it displays the following features:

1. Main Features: contains a video showing how to use the site.
2. Template: user can find the view element applied to the custom view (Button, Text View, Radio button, shape, and Edit Text).
3. Custom Java: template that we are created in java language, help the Android developers to choice between two Custom Java designs (ButtonBuzzer, ButtonHeart).
4. Help: presents how the developer can obtain information from the site.
5. Get Start: selective when the user wants to customize the Button, Textview,Edittext, Radio button, Chech box, and Shape view. and support some attribute:
6. Basic 🡪 ID, Enable, Alpha .
7. Shape 🡪Shape, Width, High, Diameter, Rib, Skew, Same Corner (Top Right Corner, Top Left Corner, Bottom Right Corner, Bottom Left Corner).
8. Text🡪Text, Text Size, Text Color, Font Family, Style, Gravity Text, Shadow Color, Shadow CentX, Shadow CentY, Shadow Radius.
9. Stroke 🡪Stroke Width, Stroke Color, Stroke Special.
10. BG Color 🡪 Solid, Gradient Type, Start Color, Center Color, End Color, Center X, Center Y, Angle, Radial Radius.
11. Padding 🡪Padding Top, Padding Bottom, Padding Left, Padding Right.
12. Ldpi, Mdpi, Hdpi, Xhdpi.
13. Generate Code.

**-Non-Functional Requirements**

Non-functional requirements place constraints on how the system will do so. The non-functional requirement elaborates a performance characteristic of the system.

**Availability:**

The system should be available, or uptime, means that they should be operational and available to use. This is specified because some systems are designed with expected downtime for activities like database upgrades and backups.

**Performance:**

Performance requirements concern the speed of operation of a system.

**Usability:**

Usability is the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of system or its component.

In XML Custom View the Usability through simplicity of the browser and any user can easily handle the browser it contains a help page that informs the user how to use it.

**Reliability:**

Reliability is the ability of a system to perform its required functions under stated conditions for a specific period of time.

In XML Custom View the Reliability through the contact section, if the user send a message, the message is delivered to the correct email address and this will be explained it in the testing part (chapter 7)**.**

**Efficiency:**

The system should utilize the resources: CPU cycles, memory, bandwidth, etc., and use them in effectively and efficient ways. Our system does not require a specific infrastructure.

**Modifiability:**

Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person-months).

**3.2.2 System Requirements**

**- Hardware Requirements**

**TABLE: 3.1 Hardware Requirements**

|  |  |
| --- | --- |
| **Specifications** | **Recommended Requirements** |
| Processor | 2.0 GHz or higher |
| Hard disk | 512 Gigabyte or higher |
| Memory | 2 GB |
| Monitor | CRT/LCD monitor |

* **Software Requirements**

**TABLE: 3.2 SOFTWARE REQUIREMENTS**

|  |  |
| --- | --- |
| **Specifications** | **Recommended Requirements** |
| Operating System | Windows XP,7,8,10 |
| Front–end | ASP.NET, HTML, CSS,Bootstrap |
| Backend | NET Framework 4.5.2(ASP.NET C#) |
| Web browser | Google Chrome, Mozilla Firefox, Microsoft Edge, Opera. |
| Server | IIS |
| Integrated Development Environment (IDE) | Microsoft Visual Studio community 2015 |

**3.2.3 Domain Requirements**

Domain requirements are important because they often reflect fundamentals of the application domain. If these requirements are not satisfied, it may be impossible to make the system work satisfactorily.

* This web application shall be designed to meet android developer's needs to design and create their views.
* This application requires web hosting service.
* The processing speed shall be fast and must be measured in unit of time.

**3.2.4 ASP.NET Web Form**

"ASP.NET simplified a number of everyday tasks and, more importantly, enabled developers to work at a higher level of abstraction. This allowed them to focus more on the core functions of the Web application rather than on common tasks around Web page design.

Based on server controls, ASP.NET allows developers to build real-world Web sites and applications with minimal HTML and JavaScript skills. The whole point of ASP.NET is productivity, achieved through powerful tools integrated in the runtime as well as the provision of development facilities, such as server controls, user controls, post back events, view state, forms authentication, and intrinsic objects. The model behind ASP.NET is called Web Forms and it was clearly inspired by the desktop Windows Forms model."[Comparing Web Forms and ASP.NET MVC]

# **CHAPTER FOUR**

# **Software Design and Implementation**

In system design phases, we will describe the system requirements, operating environment, system and subsystem architecture (Use Case Model, Sequence Diagram, Activity Diagram, Screens)

# **4.1 Use Case Diagram**

A Use Case Model describes the proposed functionality of a new system. A Use Case represents a discrete unit of interaction between a user and the system.

Each Use Case describes the functionality to be built in the proposed system, which can include another Use Case's functionality or extend another Use Case with its own behavior. Figure 4.1 case diagram. From figure 4.1, it is clear that the site is directed toward user to inform him/her and to help him/her.

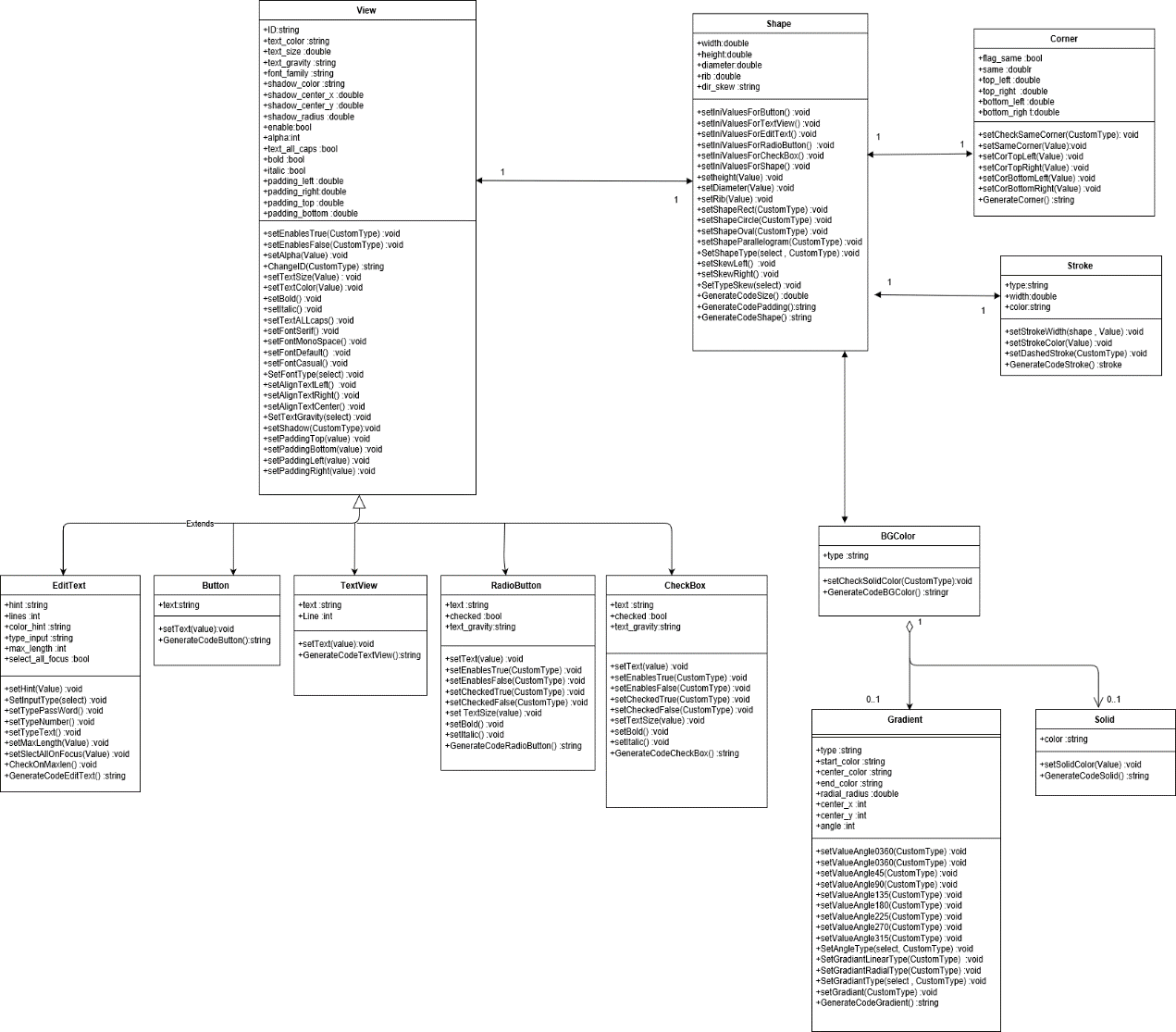


# **Figure 4.1: Use case diagram**

# **4.2 Class Diagram**

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram. Figure 4.2 class diagram

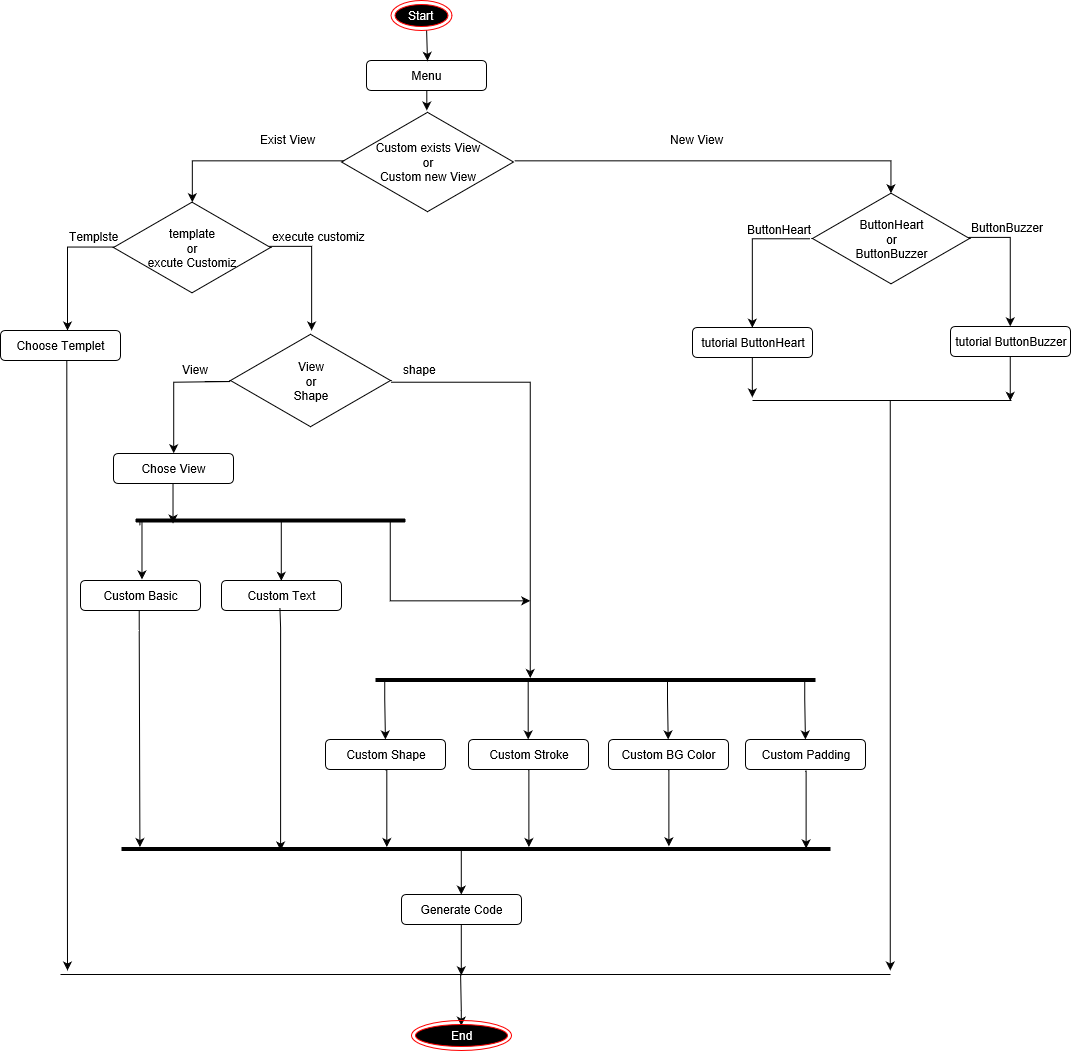


**Figure 4.2: class diagram**

# **4.3 Activity Diagram**

Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched.

Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed. Figure 4.3 Activity diagram



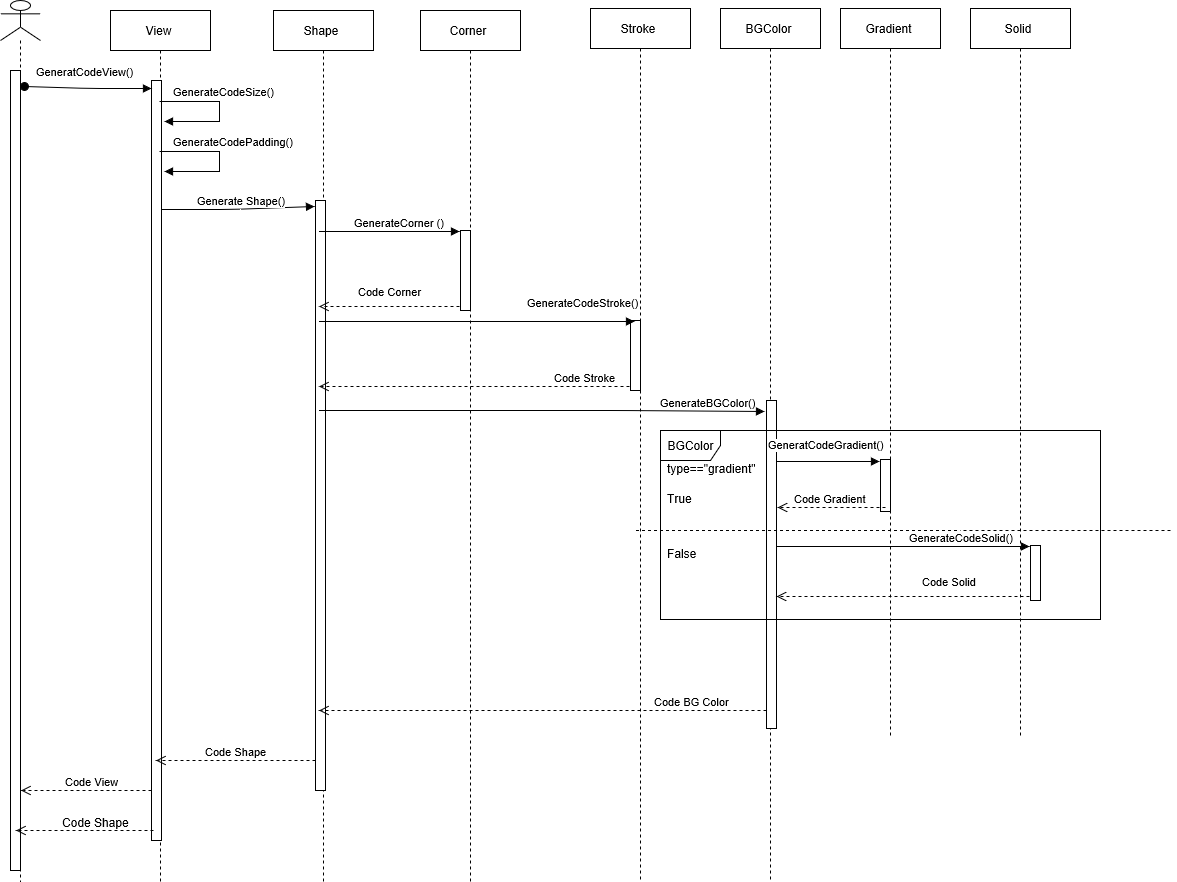
**Figure 4.3: Activity diagram**

# **4.4 Sequence Diagram**

A sequence diagram is an interaction diagram. From the name, it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one objector another.

Interaction among the components of a system is very important from implementation and execution perspective. Sequence diagram is used to visualize the sequence of calls in system to perform a specific functionality.

can be defined as the snapshot of the running system at a particular moment. Figure 4.4 Sequence diagram.



**Figure 4.4 Sequence diagram**

# **4.5 Implementation**

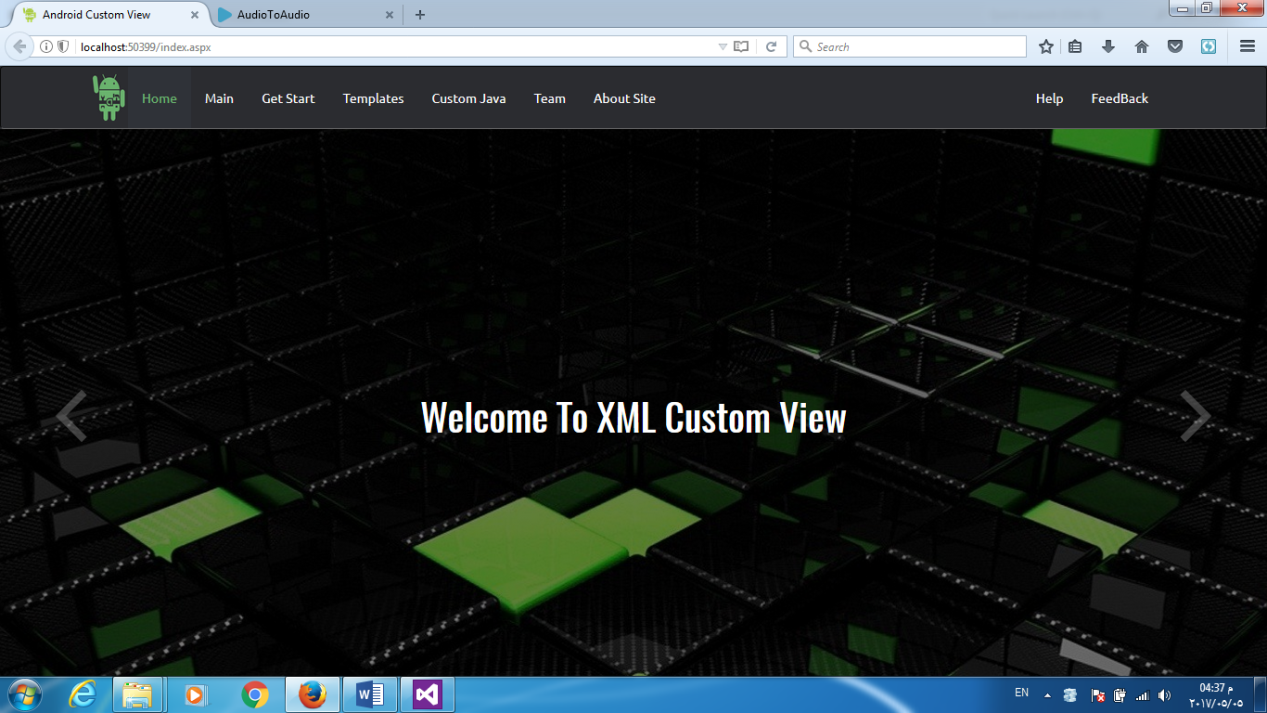
In the software implementation phase, we will show how the system, user, and domain requirements have been implemented to produce completed system. Implementation refers to the final process of moving the solution from development status to production status. On receiving system design documents, the work is divided in modules/units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

Here we will show the implementation phase by showing the final screens of the project and describing in details the functionalities that will be performed by the system through these screens.

**4.5.1 Site design System Screens**

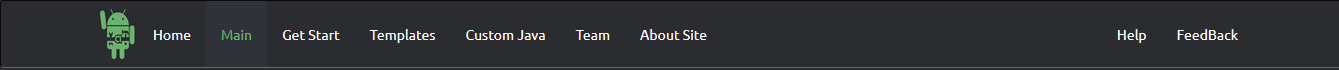
The site is composed on 9 related pages:

1. **Home page** showfigure 4.5.



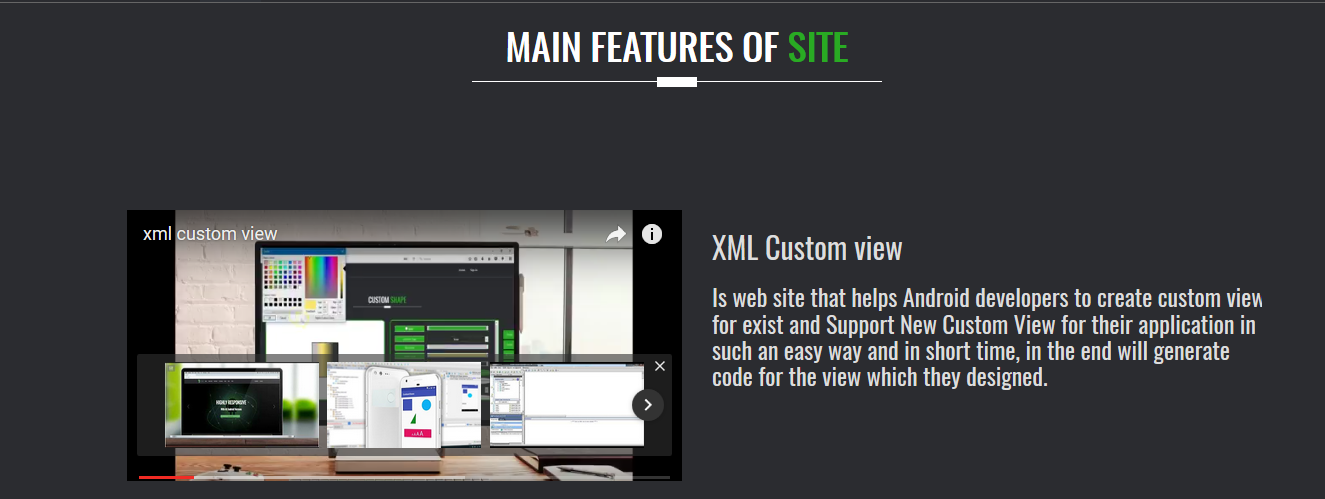
**Figure 4.5: home page**

1. **Main page** is composed on two Section the upper one contains links to all the site pages (home, get start ……) figure 4.6.



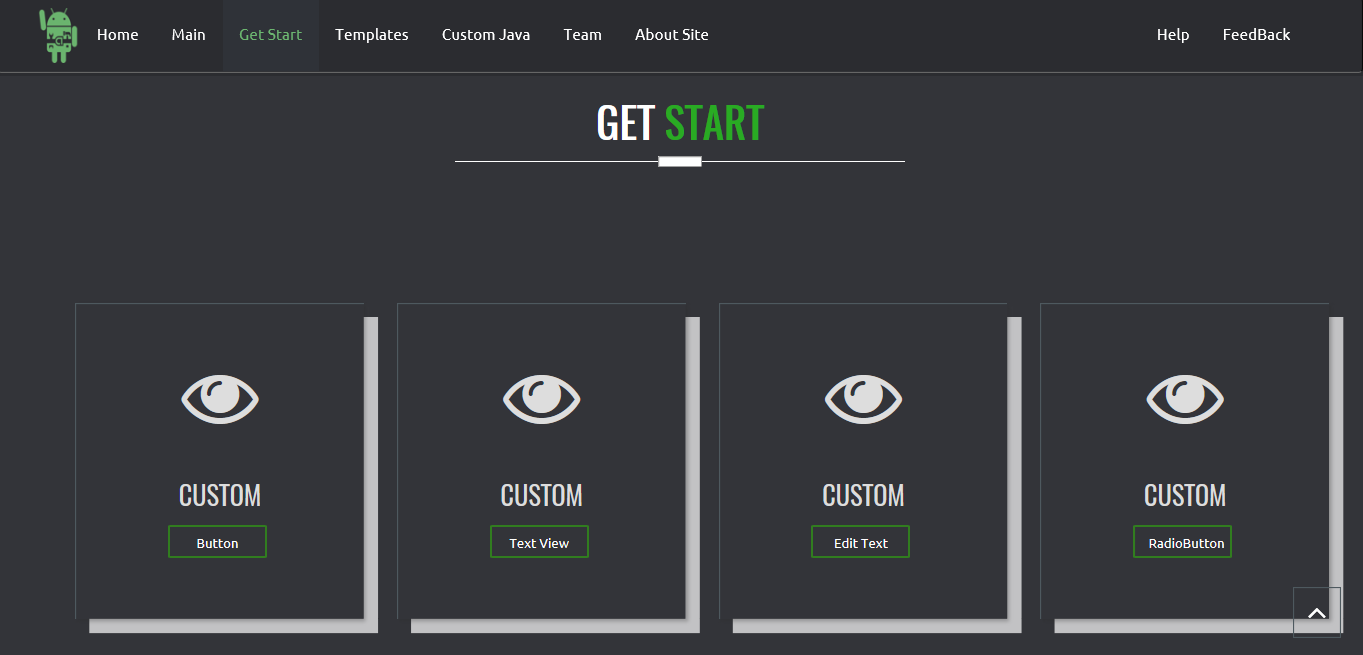
**Figure 4.6: the upper Section in the main page**

The second Section contains an introductory video that informs the visitor how to use the site, in addition to a short description of the site goal. figure 4.7



**Figure 4.7: Main Features page**

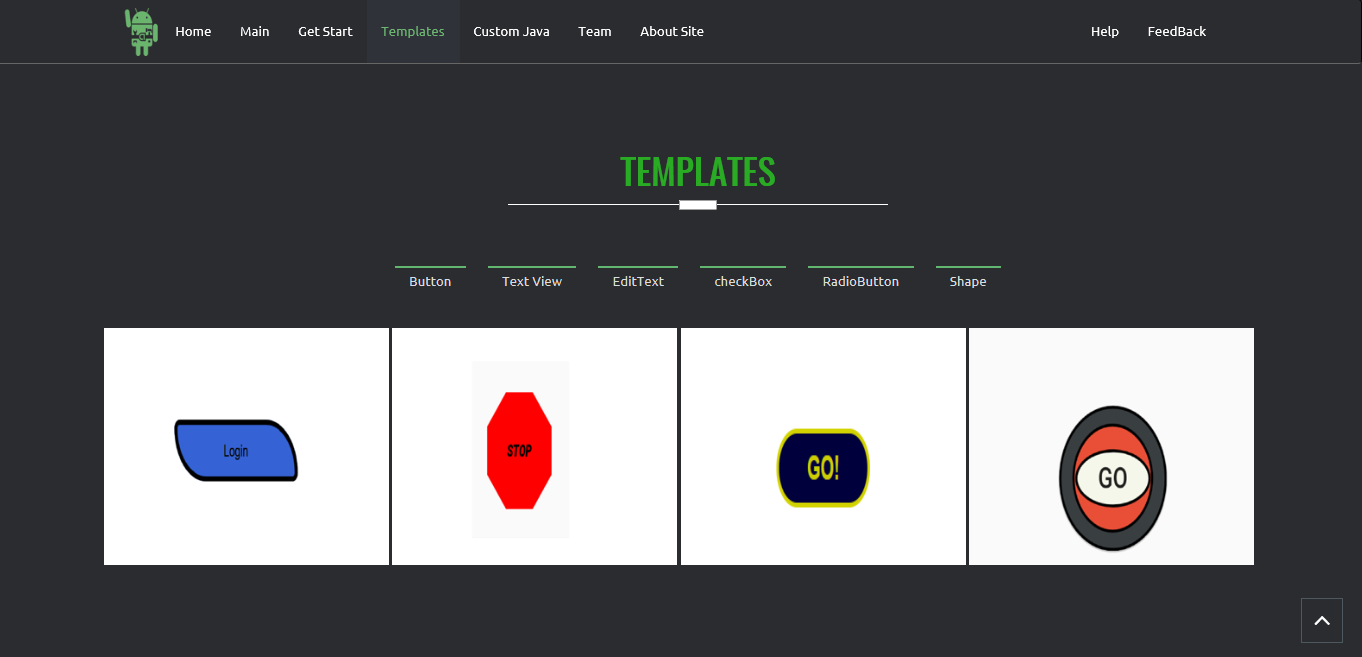
1. **Start page** is a page that is composed on 5 frames clicking on each part will move the image to another page that is the custom page. figure 4.8



**Figure 4.8: Get Start page**

**4. Template**

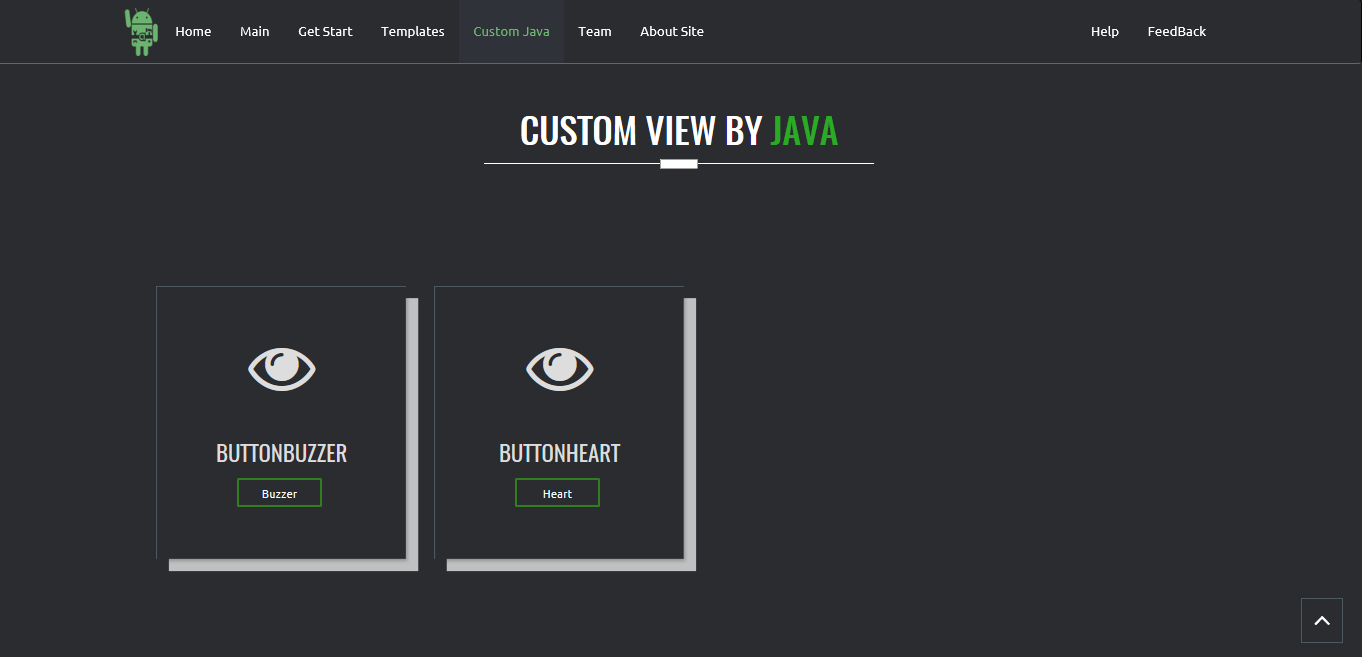
In this Section, we created a template in each of the view that can the Android Developers choose from it. Figure 4.9



**Figure 4.9: Template page**

**5. Custom Java (New View)**

Creating new view by using Java and in each of these views we provide to Android developers tutorial to help them using this view. Figure 4.11



**Figure 4.10: Custom Java page**

**6. Feedback**

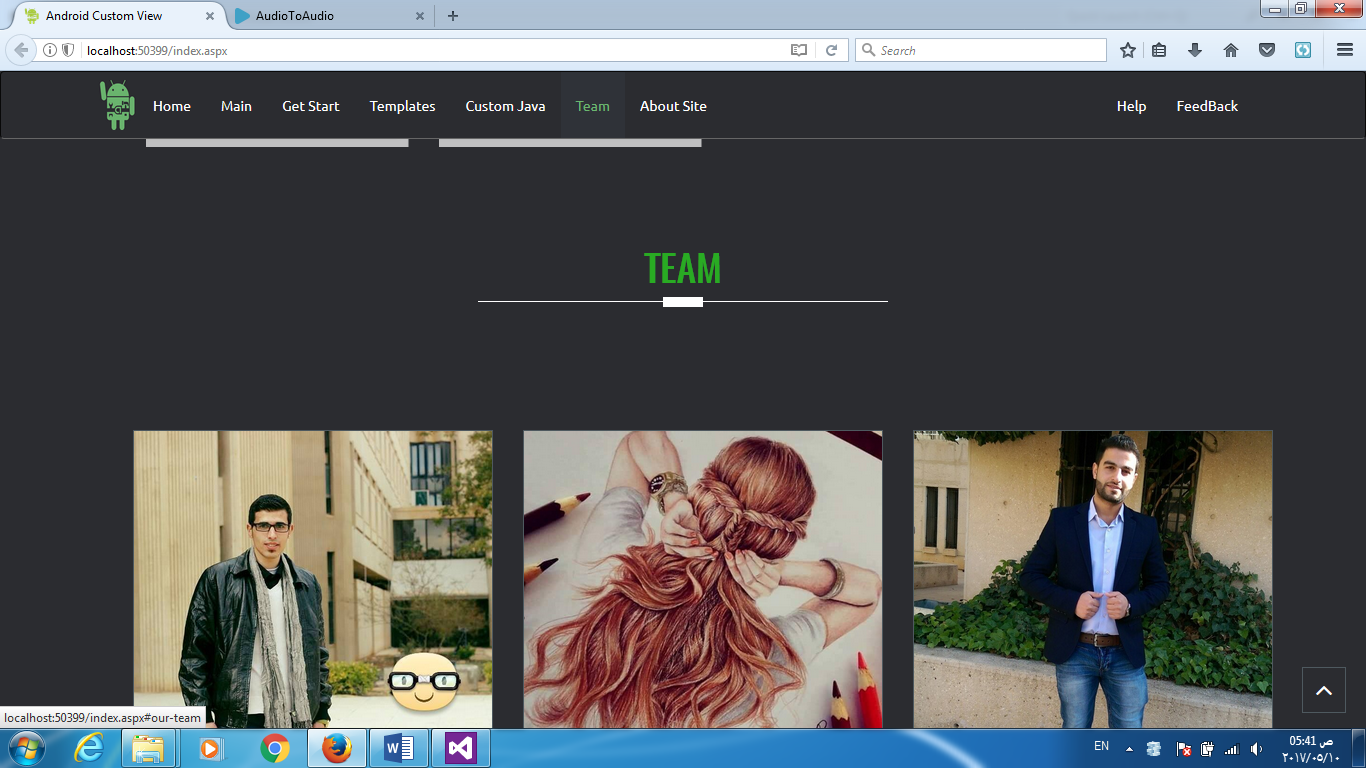
Android Developers can send feedback by using this Section. Figure 4.12



**Figure 4.11: Feedback page**

1. **Team**

This page contains information about us, project developers **Figure 4.12.**



**Figure 4.12: Team**

1. **Help**

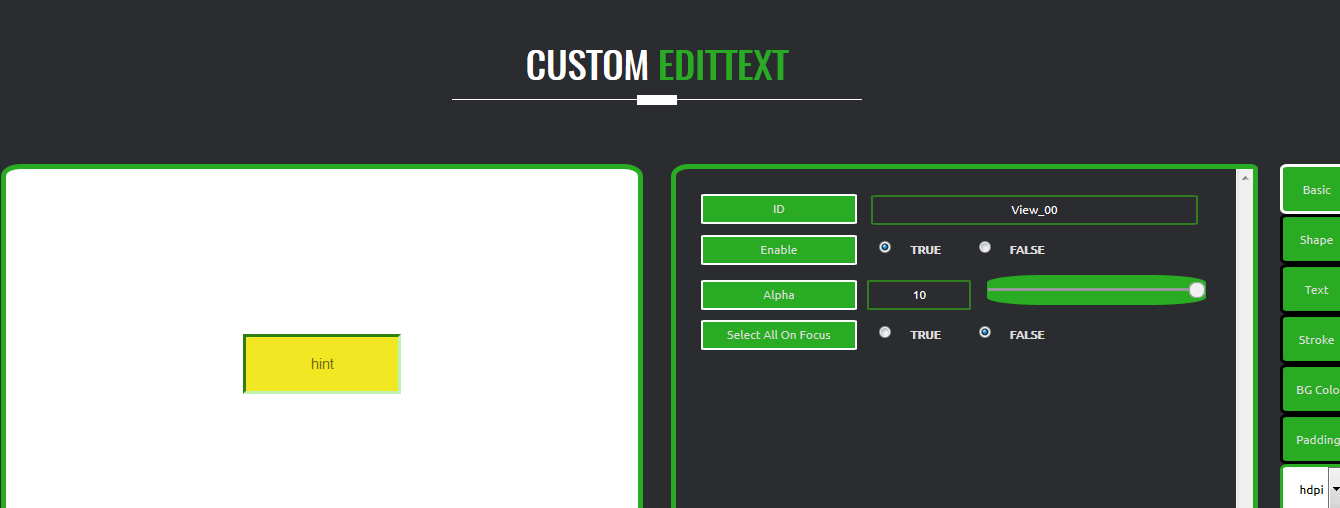
This page contains information that help the visitor to obtain information from the site.



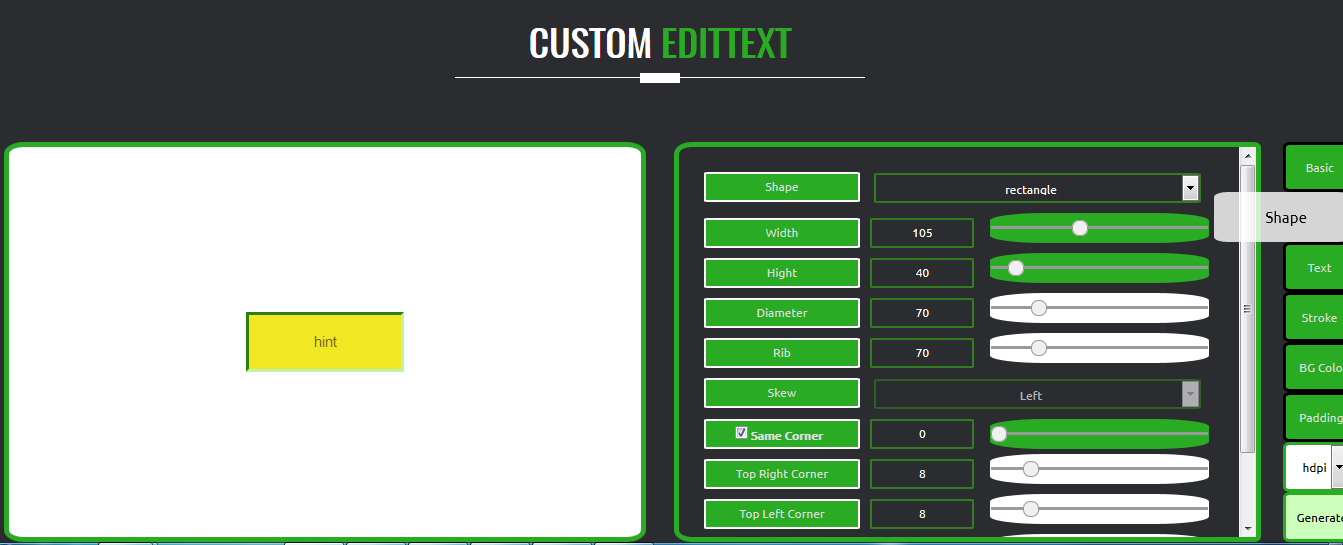
**Figure 4.13: Help**

* **On edit text custom page**,

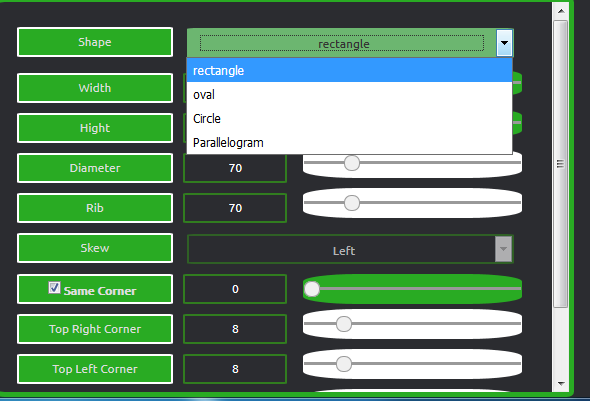
The page is divided in three parts, the leftmost one is the view of the edit text, in the middle there is the parameters changed to custom the edit text, in the rightmost one is presented the options like change the shape figure 4.14



**Figure 4.14. (a): Custom Edit Text**

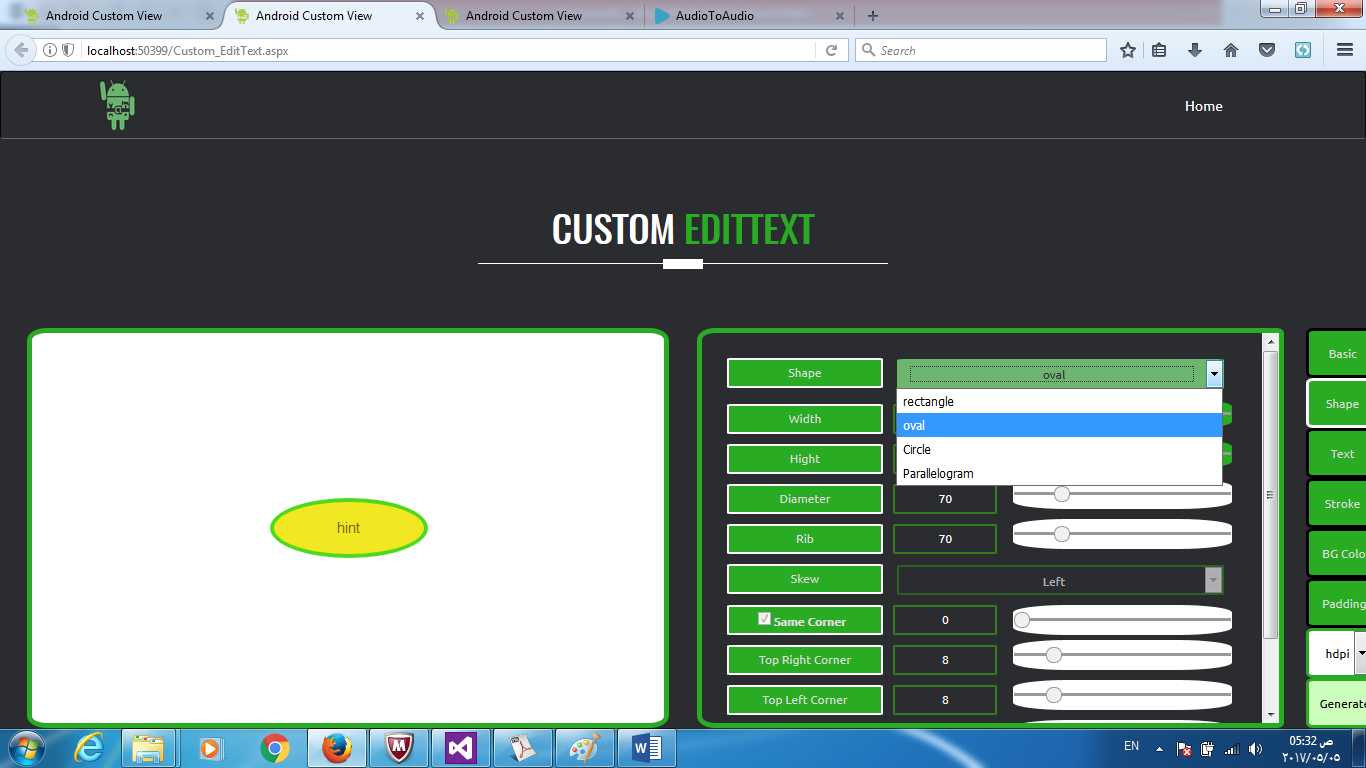


**Figure 4.14. (b): Custom EditText (shape)**



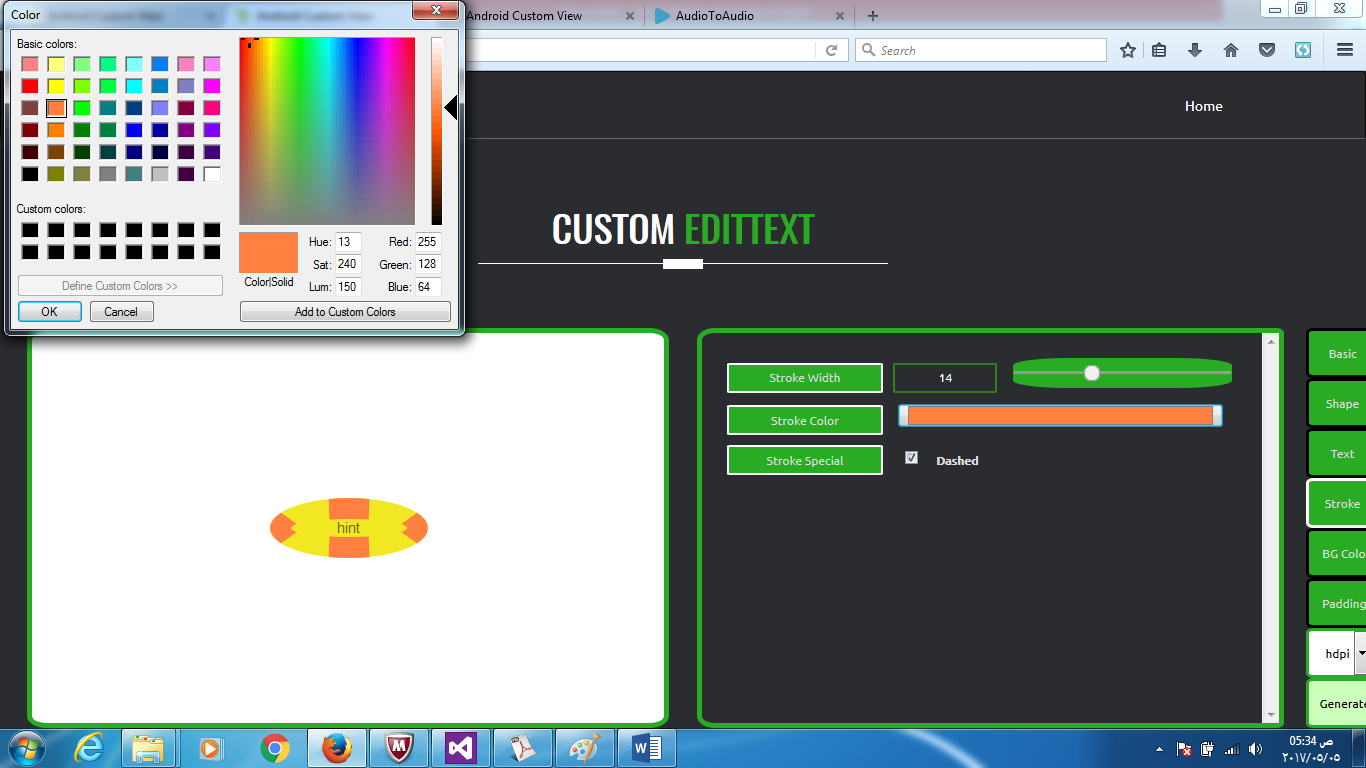
**Figure 4.15: Custom EditText without using oval**

If the user chooses the oval shape the view of edit text will be as in figure 4.15.

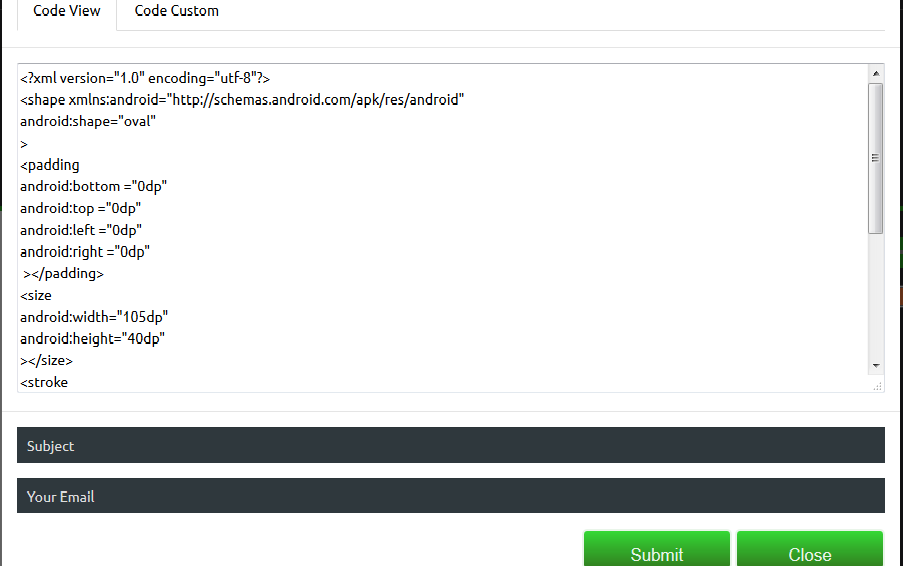


**Figure 4.16: Custom EditText with using oval**

Figure 4.16. show how the developer can change the stoke background color.

**Figure 4.17: stoke background color**

When the developer has designed the view as he wants, now the moment of generating the xml code for his application and that by clicking on generate, the last button on the right will display the user the following screen, Figure 4.17.



**Figure 4.18: generate code view**

In this figure, the developer can copy the code, or he can send it to his email by clicking on submits.

//// Code Button Submit Asp.net C#

Protected void Button\_contact\_submit\_xmlcode\_Click(object sender, EventArgs e)

{

String MSG = "Code Custom Shape \n ------------------------- \n";

MSG += TextCodeCustom.Value;

MSG += "\n------------------------- \n\n\n";

MSG += "Code View \n ------------------------- \n";

MSG += TextCodeView.Value;

MSG += "\n------------------------- \n\n\n";

MailMessage msg = new MailMessage();

msg.From = new MailAddress("XMLCustomViewSend@gmail.com");

msg.To.Add( emailCodeUser.Value.ToString() );

msg.Subject = subjectCodeUser.Value + DateTime.Now.ToString();

msg.Body = MSG.ToString();

SmtpClient client = new SmtpClient();

client.UseDefaultCredentials = true;

client.Host = "smtp.gmail.com";

client.Port = 587;

client.EnableSsl = true;

client.DeliveryMethod = SmtpDeliveryMethod.Network; client.Credentials = new NetworkCredential("XMLCustomViewSend@gmail.com", "\*\*\*\*\*\*\*\*");

client.Timeout = 20000;

try

{

client.Send(msg);

ScriptManager.RegisterClientScriptBlock(this, this.GetType(), "alertMessage", "alert('Successfully')", true);

}

catch (Exception ex)

{

ScriptManager.RegisterClientScriptBlock(this, this.GetType(),

"alertMessage", "alert('Not Successfully')", true);

}

finally

{

msg.Dispose();

}

}

**- Creating FullCustom View (ButtonBuzzer)**

Creating custom view is centered around four primary aspects that we may need to control or modify: -

1. Drawing Control the rendering of the view on screen visually by overriding the onDraw method.
2. Interaction Control the ways the user can interact with the view with the onTouchEvent and gestures.
3. Measurement Control the content dimensions of the view on screen by overriding the onMeasure method.
4. Attributes Defining custom XML attributes for your view and using them to control behavior with TypedArray.

// Code Custom View ButtonBuzzer in Java   
**public class** ButtonBuzzer**extends** CompoundButton {  
  
**private** Paint **paint**;  
**private** Path **path**;  
**private int buzzerColor**;  
**private int stroke\_buzzerColor**;  
**float width**;  
**float height**;  
  
**public** ButtonBuzzer(Context context, AttributeSetattrs, **int**defStyleAttr) {  
**super**(context, attrs, defStyleAttr);  
setupAttributes(attrs);  
**paint** = **new** Paint(Paint.***ANTI\_ALIAS\_FLAG***);  
**path** = **new** Path();  
 }  
  
**private void** setupAttributes(AttributeSetattrs) {  
*// Obtain a typed array of attributes*TypedArray a = getContext().getTheme().obtainStyledAttributes(attrs, R.styleable.***ButtonBuzzer***, 0, 0);  
  
*// Extract custom attributes into member variables***try** {  
**buzzerColor**= a.getColor(R.styleable.***ButtonBuzzer\_buzzerColor***, Color.***RED***);  
**stroke\_buzzerColor**= a.getColor(R.styleable.***ButtonBuzzer\_stroke\_buzzerColor***, Color.***GRAY***);  
 } **finally** {  
*// TypedArray objects are shared and must be recycled.*a.recycle();  
 }  
 }  
  
@Override  
**protected void** onMeasure(**int**widthMeasureSpec, **int**heightMeasureSpec) {  
**super**.onMeasure(widthMeasureSpec, heightMeasureSpec);  
  
**width** = **this**.getWidth();  
**height** = **this**.getHeight();  
 }  
  
@Override  
**public boolean**onTouchEvent(MotionEvent event) {  
  
**if** (MotionEvent.***ACTION\_UP***== event.getAction())  
**this**.setChecked((**this**.isChecked())? **false** :**true**);  
  
 invalidate();  
requestLayout();   
  
*//postInvalidate();***return true**;  
 }

2

3

4

@Override  
**protected void** onDraw(Canvas canvas) {  
**super**.onDraw(canvas);  
  
**width** = **this**.getWidth();  
**height**= **this**.getHeight();  
  
**float** dis = (Math.*min*(**width** , **height**)/10);  
  
**float** TL2x = (**width**/10) + dis ;  
**float** TL2y = (**height**/10) - dis;  
  
**float** TL1x = (**width**/10) - dis ;  
**float** TL1y = (**height**/10) + dis;  
  
**float** BL1x = TL1x ;  
**float** BL1y = (**height** - (**height**/10)) - dis;  
  
**float** BL2x = TL2x ;  
**float** BL2y = (**height** - (**height**/10)) + dis;  
  
**float** BR1x = (**width** - (**width**/10)) - dis ;  
**float** BR1y = BL2y;  
  
**float** BR2x = (**width** - (**width**/10)) + dis ;  
**float** BR2y = BL1y;  
  
**float** TR2x = BR2x;  
**float** TR2y = TL1y ;  
  
**float** TR1x = BR1x;  
**float** TR1y = TL2y ;  
  
**path**.moveTo(TL2x , TL2y);  
**path**.lineTo(TL1x , TL1y);  
**path**.lineTo( (**width**/2) - dis , **height**/2);  
**path**.lineTo(BL1x , BL1y);  
**path**.lineTo(BL2x , BL2y);  
**path**.lineTo(**width**/2 , (**height**/2) + dis);  
**path**.lineTo(BR1x , BR1y);  
**path**.lineTo(BR2x , BR2y);  
**path**.lineTo((**width**/2)+ dis , **height**/2);  
**path**.lineTo(TR2x , TR2y);  
**path**.lineTo(TR1x , TR1y);  
**path**.lineTo(**width**/2 , (**height**/2)- dis );  
**path**.lineTo(TL2x , TL2y);  
  
**paint**.setShader(**null**);  
**paint**.setColor(**buzzerColor**);  
  
**if** (**this**.isChecked()){  
**paint**.setStyle(Paint.Style.***FILL***);  
canvas.drawPath(**path**, **paint**);  
 }  
  
**paint**.setColor(**stroke\_buzzerColor**);  
**paint**.setStyle(Paint.Style.***STROKE***);  
**paint**.setStrokeWidth(dis/3);  
canvas.drawPath(**path**, **paint**);  
 }  
  
**public int**getBuzzerColor() {  
**return buzzerColor**;  
 }  
  
**public void** setBuzzerColor(**int**buzzerColor) {  
**this**.**buzzerColor**= buzzerColor;  
 invalidate();  
requestLayout();  
 }  
  
  
**public int** getStroke\_buzzerColor() {  
**return stroke\_buzzerColor**;  
 }  
  
**public void** setStroke\_buzzerColor(**int**stroke\_buzzerColor) {  
**this**.**stroke\_buzzerColor**= stroke\_buzzerColor;  
 invalidate();  
requestLayout();  
 }  
  
}

1

**CHAPTER FIVE**

**Results and Discussion**

**5.1 Result**

**5.1.1 Expected Result**

In this project, we were interested in developing a site for the programmers / developer’s android to help them to design their view in a professional manner without the need for a great time. The main idea, were to create a website that is easy to use and efficient at the same time and can be accessed any time without constrains on the time and location.

When studying the relevant works, we found that the best site supports 26 shape & Button Attribute and All related works that support only button but one application (Dev Tools - Style Splash PRO) support TextView but it is paid his price (1.11 $). We seek in this project to support at least 35 shape & Button Attributes, And support the largest number of View such as (Button, TextView ,EditText ,Radio Button , Check Box).

It is expected to allow site XML View Custom Ready to provide templates and provide some custom view (Button, TextView ,Radio Button , Check Box, Edit Text ) , New custom view (Buzz view , Heart View), After selecting the user and creating the appropriate design for him, the code is taken by click on the button Generate code.

In the design, it is expected to support some shapes such as (square - triangle - oval - parallelogram - rectangle)

**5.1.2 Actual Results**

The actual results of XML Custom View: -

1- The XML Custom View will offer some ready-made templates.

2-Supports 43 shape & button Attribute and supports a max number of view (Button, TextView ,Radio Button , Check Box, Edit Text) .

3-Modify the look and feel of the View by choosing the appropriate view such as custom java (Button Buzz, Button Heart), custom view (Button, TextView ,EidtText, Radio Button, Check Box ,shape) .Then use some of the available tools ,to modify the view.

4-There is an interaction between the site and the user through the user is using the tools and make some adjustments to the view and shows the site result immediately and also through all the adjustments to the user by clicking on the code generation, the site provides the code to the user immediately.

5-Some shapes are supported in design such as (circle - oval - parallelogram – rectangle).

**5.2 Discuss**

After discussing the expected and actual results, the difference between them is clear. We have achieved Most of the expected work, such as

It was expected to support 35 features we supported 43 shape & button attributes and has been supported as many possible view and create a Custom Java as expected we will show that in (Table 5.1, Table 5.2, Table 5.3).

In (Table 5.4) shows the difference between our project and the related work, showing the difference between the number of supported attributes and how many views are supported and if support custom java or not supported it, we will note that only our project (XML Custom View) has supported a large number of attributes and most of the view and support of two components of custom java.

But we encountered some of the problem in designing shapes because of the limited language of CSS such as shape of triangle and pentagram.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tag** | **Attribute** | **Android Drawable Generator** | **Android Button Maker** | **Drawable Designer** | **XML Button Maker CM** | **Android Shape Generator** | **Dev Tools style splash** | **XML Custom View** |
| Size | width | × | √ | √ | √ | × | × | √ |
| height | × | √ | √ | √ | × | × | √ |
| Corners | radius | √ | √ | √ | × | × | √ | √ |
| bottom left radius | √ | √ | √ | √ | × | √ | √ |
| bottom right radius | √ | √ | √ | √ | × | √ | √ |
| top left radius | √ | √ | √ | √ | × | √ | √ |
| top right radius | √ | √ | √ | √ | × | √ | √ |
| Stroke | width | √ | √ | √ | √ | × | √ | √ |
| color | √ | √ | √ | √ | × | √ | √ |
| dash Gap | × | × | √ | √ | × | × | √ |
| dash width | × | × | √ | √ | × | × | √ |
| Solid | color | √ | √ | √ | √ | × | √ | √ |
| Gradient | type | √ | × | × | × | × | × | √ |
| star color | √ | √ | √ | √ | √ | √ | √ |
| center color | √ | √ | √ | √ | × | √ | √ |
| end color | √ | √ | √ | √ | √ | √ | √ |
| center x | √ | √ | √ | √ | √ | √ | √ |
| center y | × | × | √ | √ | √ | √ | √ |
| linear angle | √ | √ | √ | √ | × | × | √ |
| radial radius | × | × | × | × | × | × | √ |
| Padding | top | √ | √ | √ | × | √ | √ | √ |
| bottom | √ | √ | √ | × | √ | √ | √ |
| left | √ | √ | √ | × | √ | √ | √ |
| right | √ | √ | √ | × | √ | √ | √ |

**Table 5.1: attributes Shape in related work & XML Custom View**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tag** | **Attribute** | **Android DrawableGenerator** | **Android Button Maker** | **Drawable Designer** | **XML Button Maker CM** | **Android Shape Generator** | **Dev Toolsstyle splash** | **XML Custom View** |
| Button | id | × | × | × | × | × | × | √ |
| text | √ | √ | × | × | × | × | √ |
| text color | √ | √ | × | √ | × | √ | √ |
| text size | √ | √ | × | √ | × | √ | √ |
| text style bold | × | × | × | × | × | × | √ |
| text style italic | × | × | × | × | × | × | √ |
| gravity | × | × | × | × | × | × | √ |
| font family | × | × | × | × | × | × | √ |
| layout-width | × | √ | √ | √ | × | √ | √ |
| layout-height | × | √ | √ | √ | × | √ | √ |
| enable | × | × | × | × | × | × | √ |
| alpha | × | × | × | × | × | × | √ |
| background | √ | √ | √ | √ | × | √ | √ |
| text all caps | × | × | × | × | × | × | √ |
| radius shadow | √ | × | × | × | × | × | √ |
| delta x shadow | √ | × | × | × | × | × | √ |
| delta y shadow | √ | × | × | × | × | × | √ |
| color shadow | √ | × | × | × | √ | × | √ |
| Shape | rectangle | rectangle | rectangle, oval, line, ring | Rectangle, oval | rectangle | rectangle | Rectangle, oval ,circle , parallelogram |

**Table 5.2: attributes Button in related work & XML Custom View**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tag** | **Attribute** | **Android DrawableGenerator** | **Android Button Maker** | **Drawable Designer** | **XML Button Maker CM** | **Android Shape Generator** | **Dev Toolsstyle splash** | **XML Custom View** |
| Text View | id | × | × | × | × | × | × | √ |
| text | × | × | × | × | × | × | √ |
| text color | × | × | × | × | × | √ | √ |
| text size | × | × | × | × | × | √ | √ |
| text style bold | × | × | × | × | × | × | √ |
| text style italic | × | × | × | × | × | × | √ |
| gravity | × | × | × | × | × | × | √ |
| font family | × | × | × | × | × | × | √ |
| layout-width | × | × | × | × | × | √ | √ |
| layout-height | × | × | × | × | × | √ | √ |
| background | × | × | × | × | × | √ | √ |
| lines | × | × | × | × | × | × | √ |
| textallcaps | × | × | × | × | × | × | √ |
| radius shadow | × | × | × | × | × | √ | √ |
| delta x shadow | × | × | × | × | × | √ | √ |
| delta y shadow | × | × | × | × | × | √ | √ |
| color shadow | × | × | × | × | × | √ | √ |

**Table 5.3: attributes Text View in related work& XML Custom View**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **XML Custom View** | **Dev Toolsstyle splash** | **Android Shape Generator** | **XML Button Maker CM** | **Drawable Designer** | **Android Button Maker** | **Android DrawableGenerator** |  |
| 24 | 17 | 8 | 17 | 22 | 19 | 18 | Shape |
| 19 | 6 | 2 | 6 | 4 | 7 | 9 | Button |
| 17 | 9 | × | × | × | × | × | Text View |
| 19 | × | × | × | × | × | × | Edit Text |
| 17 | × | × | × | × | × | × | Radio Button |
| 17 | × | × | × | × | × | × | Check Box |
| We can found(2) new View | Not Found | Not Found | Not Found | Not Found | Not Found | Not Found | Custom java |

**Table 5.4: Comparison between related work & XML Custom View**

**CHAPTER SIX**

# **Testing**

# **6.1 Black-Box Testing:**

## 6.1.1 Dynamic black-box testing

Dynamic black-box testing is testing without having an insight into the details of the underlying code.

* + Dynamic, because the program is running
  + Black-box, because testing is done without knowledge of how the program is implemented.
  + Sometimes referred to as behavioral testing.
  + Requires an executable program and a specification (or at least a user manual).
  + Test cases are formulated as a set of pairs

## 6.1.2 Test Data and Test Cases

* Test data: Inputs which have been devised to test the system.
* Test cases: Inputs to test the system and the predicted outputs from these inputs if the system operates according to its specification.

## 6.1.3 Test-to-pass:

* + assures that the software minimally works,
  + does not push the capabilities of the software,
  + applies simple and straightforward test cases,
  + Does not try to “break” the program.

## 6.1.4 Test-to-fail:

* Designing and running test cases with the sole purpose of breaking the software.
* Strategically chosen test cases to probe for common weaknesses in the software.

## 6.1.5 Characteristics of Black-box testing:

* + Program is treated as a black box.
  + Implementation details do not matter.
  + Requires an end-user perspective.
  + Criteria are not precise.
  + Test planning can begin early.

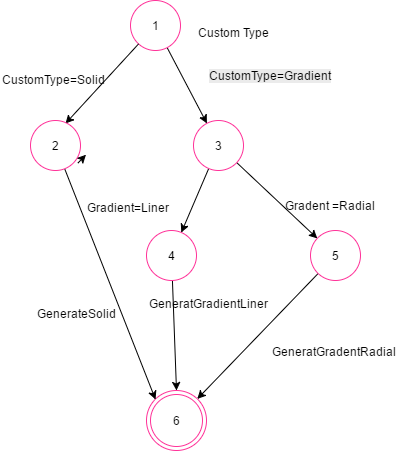
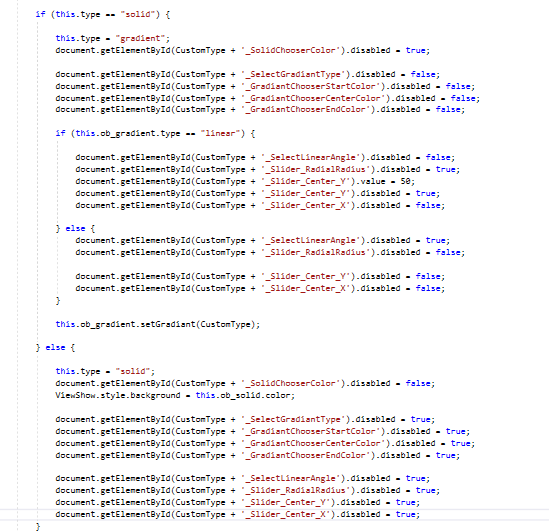
# **6.2 White-box Testing:**

## 6.2.1 Data-Control Testing:

* Control-flow testing is a structural testing strategy that uses the program’s control flow as a model.
* Control-flow testing techniques are based on selecting a set of test paths through the program.
* The set of paths chosen is used to achieve a certain measure of testing thoroughness.

## 6.2.2 Data-flow testing

* Data-flow testing uses the control flow graph to explore the unreasonable things that can happen to data.(**Figure**7.1)
* Consideration of data-flow anomalies leads to test path selection strategies that fill the gaps between complete path testing and branch or statement testing.
* Data-flow testing is the name given to a family of test strategies based on selecting paths through the program’s control flow in order to explore sequences of events related to the status of data objects.



**Figure 6.1: Graph**

**Table 7.1: Button attribute testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Page** | **attribute** | **Test Data** | **Actual** |
| **Custom Button** | **ID** | **Btn** | **√** |
| **Enable** | true | √ |
| **Alpha** | 8 | √ |
| **Shape** | rectangle | √ |
| **Width** | 170 | √ |
| **Hight** | 46 | √ |
| **Diameter** | inactive | √ |
| **Rib** | inactive | √ |
| **Skew** | inactive | √ |
| **Same Corner** | inactive | √ |
| **Top Right Corner** | 19 | √ |
| **Top Left Corner** | 41 | √ |
| **Bottom Right Corner** | 32 | √ |
| **Bottom Left Corner** | 8 | √ |
| **Text** | Go | √ |
| **Text Size** | 13 | √ |
| **Text Color** | Blue#3C89E2 | √ |
| **Font Family** | serif | √ |
| **Style** | Italic | √ |
| **Gravity Text** | center | √ |
| **Shadow Color** | Yellow#F3F763 | √ |
| **Shadow CentX** | 4 | √ |
| **Shadow CentY** | 1 | √ |
| **Shadow Raduis** | 4 | √ |
| **Stroke Width** | 3 | √ |
| **Stroke Color** | Read | √ |
| **Stroke Special** | Dashed | √ |
| **Solid** | False | √ |
| **Gradient Type** | True | √ |
| **Start Color** | Read #F78263 | √ |
| **Center Color** | Green #3FD246 | √ |
| **End Color** | Pink #F041E2 | √ |
| **Center X** | 25 | √ |
| **Center Y** | 50 | √ |
| **Angle** | 255 | √ |
| **Radial Radius** | inactive | √ |
| **Padding Top** | 30 | √ |
| **Padding Bottom** | 20 | √ |
| **Padding Left** | 20 | √ |
| **Padding Right** | 12 | √ |

**Table 7.2: BlackBox Test**

|  |  |
| --- | --- |
| **Xml Custom View** | **Android Studio** |
|  |  |
|  |  |
|  |  |

# **CHAPTER SEVEN**

**Conclusion and Future Works**

**7.1 Conclusion**

In this project, we have created a site (XML Custom View) that will help developers / programmer’s android to design their applications in a special and distinctive in an easy, fast and sufficient. In this program all functions of the user are executed such as: select a view, select a tool (Basic, Shape, Text, Stroke, BG Color, Padding, and Size of screen), Modify the properties. This work was hard, complex and interest in the same time by completing this project the developers / programmer’s android will not suffer from the complexities of the process of creating a custom view as desired, it will be performed in a short time.

In chapter one we have defined our objectives as follow:

1. To design view in an android application in soft way.
2. To simplify the view design for android developers.
3. To shorten the time for android developers.
4. To make the first, second and third points more flexible and simple.

Now we have terminated the work in our project we can said that we have realized our objectives where we have developed a web site that helps android developers in designing a different android application interface in an easy and flexible manner and it does not require a long time or a specific skill.

Our goal were creativity, usability and novelty and we have designed our web site to help the android developer to use his creativity in his application, our site is easy and simple to use and can be used by a large part of the android developers because the information presented in it are simple and useful. about novelty, in our project we have developed two new views are (Buzzer, heart) that can be used and customize by the developer.

**7.2 Future Work**

In the near future, more functionalities will be added to this system, these functions will be like:

1. One of our tasks in Future Work is to create screens or layouts that: setup a root ViewGroup, start adding UI elements (aka widgets) and other ViewGroups until we have the desired layout.
2. Support more than new Custom View.
3. Support multi state: all smart shapes, text captions, objects, and media can have multiple states

**References**

1. Wei-MengLee." beginning Android Application Development " .19 April 2011. Wrox; 1 edition. **ISBN-10:** 9781118017111. [Accessed 20 03 2017].
2. AndroidSecurityBulletin–February 2017. [Online].Available: <https://source.android.com/security/bulletin/2017-02-01>.[Accessed 27 03 2017].
3. GoDoc.Retrieved March 12, 2017. [Accessed 02 04 2017].
4. AndroidTutorial.[Online].Available: <https://www.tutorialspoint.com/android/android_application_components.htm>. [Accessed 10 04 2017].
5. JavaDebuggingwithEclipseTutorial[Online].Available: <http://www.vogella.com/tutorials/AndroidCustomViews/article.html>. [Accessed 19 04 2017].
6. XML. [Online].Available:<https://www.tutorialspoint.com/xml/>[Accessed 26 04 2017].
7. (UI – GUI) MSCOM March 12, 2007. [Online].Available: <https://blogs.technet.microsoft.com/mscom/2007/03/12/the-gui-versus-the-command-line-which-is-better-part-1/>.[Accessed 29 04 2017].
8. (Android Layout Types). [Online].Available: <https://www.tutorialspoint.com/android/android_user_interface_layouts.htm>. [Accessed 2 05 2017].
9. (AndroidUIControls). [Online].Available: <https://developer.android.com/guide/topics/ui/controls.html>. [Accessed 08 05 2017].
10. MVC, "Comparing Web Forms and ASP.NET MVC," [Online]. Available: https://msdn.microsoft.com/en-us/magazine/dd942833.aspx. [Accessed 2016 5 11].