CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test 1 – Sep. 2019

Sub:	Mobile Applications							Sub Code:	17MCA53
Date:	23/09/2019	Duration:	90 min's	Max Marks:	50	Sem:	V	Branch:	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

		MARKS	OBE	
			CO	RBT
	PART I			
1	What do you understand by mobile information design and mobile platform? Explain it.	[10]	CO1	L1
	OR			
2	Explain the effective use of screen real estate and features of mobile application	[10]	CO1	L2
	PART II			
3	What are the components of android applications? Explain with example.	[10]	CO2	L1,L2
	OR			
4	What are the preliminary costs involved in mobile application development	[10]	CO1	L1
	PART III			
5	What is android? Explain the android architecture with its features and diagram	[10]	CO2	L2
	OR			
6	Describe the basic views in android with a suitable code snippet.	[10]	CO2	L2
	PART IV			
7	What is a fragment? Explain the lifecycle of a fragment	[10]	CO2	L2
	OR			
8	Explain the various information design tools of mobile interface design.	[10]	CO1	L2
	PART V			
9	Briefly discuss the Gestalt's principles.	[10]	CO1	L1
	OR			
10	Describe the anatomy of android application.	[10]	CO2	L2

CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test 1– Sept. 2019

Sub:	Mobile Applications					Sub Code:	17MCA53	Branch:	MCA
Date:	23/09/2019	Duration:	90 min's	Max Marks:	50	Sem	V		OBE

Q1) What do you understand by mobile information design and mobile platform? Explain it.

Mobile devices offer an exciting space to design information, fitting personalized and realtime data into tightly-constrained screens. But keep audience goals in mind when crafting an application, because mobile devices are not generally used for extensive browsing or complex searches. Various mobile design patterns are discussed hereunder.

1.6.1 Information Display

People identify signals, interpret the meaning of these signals, determine the goal according to these

interpretations, and then carry out an action until the goal is reached.

o For example, a microwave has a simple display. When the timer alerts us the popcorn is done.

o Overly detailed designs do not suit mobile users, who are often micro-tasking, influenced by urgent and new

surroundings, and looking for a quick fix for boredom.

.6.2 Design Patterns

o A design pattern recycles and repurposes components, reusing the best ideas.

o More than time efficiency, patterns have been refined by use.

o Look to the user and the purpose of an individual design above any best practices.

Navigation

Good design makes it clear how users can move through and use application

features.

Annunciator Panel

– Fixed Menu

– Expandable Menu

– Scroll

Notifications and Feedback

i. Annunciator Panel

o An annunciator panel, seen at the top of Figure 1.6, gives information on the state of a mobile device.

o Developers can modify or suppress the annunciator panel — which lists the hardware features such as

network connection and battery power — within an application.

o Because the information in this area is only notifications, application users will not usually have any direct interaction with the annunciator panel.

ii. Fixed Menu

A menu that remains fixed to the viewport as users roam content is useful in many situations:

- When users need immediate access to frequently selected functionality on multiple screens
- When a reveal able menu is already used on the same screen
- When a lack of controls, conflict with key interactions, or low discovery makes a reveal able menu a poor choice.
- Because fixed menus are always accessible , users can interact with page content as needed; keep in mind the small screen real estate, and limit any fixed menus to the absolute necessities.

Scroll:

As in the case of a reveal able menu giving additional functionality, there will often be more content on a screen than can be seen in the device viewport. o It is best to limit scrolling, limiting application screens to the size of the viewport whenever possible. o Only in-focus items should be able to scroll. o Scrolling must occur both horizontally and vertically. o An endless list of information breaks large data sets into manageable,consumable sizes within the viewport. o An expandable list, shown in Figure 1.9, reveals additional, valuable content

1 Android

- Android has a diverse ecosystem, with fewer institutionalized restrictions and a wider variety of mobile devices than other popular systems.
- They are strong competitor in the mobile market, but the flexibility of Android design can introduce new issues.
- Development of the Android operating system is led by Google http://developer.android.com/guide/practices/ui_guidelines/index.html

Interface Tips

– Get started on Android application design with these hints.

- Android convention is to place view-control tabs across the top, and not the bottom, of the screen.
- Use the main application icon for temporal, hierarchical navigation, instead of a “back” button and main icon link to the home screen.
- Don’t mimic user interface elements or recycle icons from other platforms.
- For instance, list items should not use carets to indicate deeper content.
- Parallax scrolling is common in Android applications.
- Android development can extend to home-screen “widget” tools.

Accessibility

- Google provides guidelines and recommendations, such as testing with the often-preinstalled and always-free Talkback. Accessibility design guidelines are listed on the Android Developer website (http://developer.android.com/guide/topics/ui/accessibility/index.html), and further discussed by the Google “Eyes Free” project (http://eyesfree.googlecode.com/svn/trunk/documentation/android_access/index.html).

2 iOS

- Apple maintains strict design standards, which are detailed and updated online. iOS
- interface documentation and general mobile design strategies are available from Apple, including design strategies and case studies, at http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html

Interface Tips

- -Apple can reject an application from the official App Store because of design problems. Follow the current guidelines closely, starting with these tips:
- □ iPhone users generally hold from the bottom of the device, so main navigation items should be in reach of user thumbs.
- □ Target areas for controls should be a minimum of 44 x 44 points.
- □ Support standard iOS gestures, such as swiping down from the top to reveal the Notification Center.
- □ Larger iPad screens are great for custom multi-finger gestures, but nonstandard gestures should never be the only way to reach and use important features.

Accessibility

- □ Accessible touch and gestural controls are available on the iPad and later generation iPhones;
- □ Screen magnification and colour contrast adjustments are also available.

3 BlackBerry OS

- BlackBerry OS is often the mobile device of choice in or corporate environments.
- BlackBerry includes native support of corporate emails; and runs on many devices with hard keypads. Which is favoured by users with accessibility issues as well as late adopters to touch-screen interfaces.

Interface Tips

- Link common tasks to the BlackBerry track pad according to standard actions:

- □Press the track pad: Default option, like revealing the menu
- □Press and hold track pad: Activate available pop-up box
- □Press track pad and select Shift: Highlight content
- □Press track pad and select Alt: Zoom
- □ Move finger along track pad: Cursor or mouse will move accordingly

Accessibility

- BlackBerry mobile devices include text-based push-delivery messages, closed captions on multimedia content, and hearing-aid compatibility for hearing accessibility issues. Low-vision users can use the Clarity theme and other screen adjustments, and benefit from tactile keyboards. Predictive text and AutoText aid users with mobility and cognitive issues. Best practices and device capabilities are maintained online at http://docs.blackberry.com/en/

4 Windows Phone7

- Developed by Microsoft, Windows Phone 7 (WP7) is a currently smaller contender, focused on consumer markets.
- Using the "Metro" theme, features are divided into "Live Tiles" that link to applications.
- Six dedicated hardware buttons (back, start, search, camera, power, and volume), at least 4 GB of Flash memory, and Assisted GPS.

Interface Tips

- Windows Phone 7 interfaces are minimalist,
- □ Using empty space to lend clarity to the application. WP7 uses movement over gradients for on-screen elements to immerse users in the application experience. Users will enter a WP7 application from a "tile," which can display dynamic and real-time information.
- □ Tile images should be in the PNG format, 173 pixels 173 pixels at 256 dpi.eaving focus. Do not use a "back" button to navigate back the page stack. Panorama controls slide horizontally through panes, and pivot controls list panes users can visit.

- Uniform Page Shuffle presents non-hierarchical information users can shuffle through;
  - “leaf-blowing turn” flips content area into focus,
  - Scattering and tilting tiles leaving focus.

Accessibility

- WP7 devices include many standard accessibility features, such as color and
- Contrast adjustment to themes for low-vision users. Many, but not all, devices
- are compatible with TTY, TDD, and hearing aids.
- Learn more about the basics of WP7 accessibility at
- http://www.microsoft.com/

5 Mobile Web Browsers

- If a mobile application sends users to a website, that website should be optimized for mobile browsers.
- Similarly, mobile web applications should follow key mobile design methods.
- A great resource for design best practices for mobile web browsers is published by the W3C.

Interface Tips

Few quick tips to get started:

- Test for a consistent experience: when websites are accessed from a variety of mobile browsers.
- Provide minimal navigation: at the top of the page, and use consistent navigation mechanisms.
- Do not change or refresh the current window: or cause pop-ups, without informing the user and providing the means to stop it.
- Limit content: what the user has requested, and what the user’s device can display by avoiding large image files.
- Specify default input formats : when possible, provide preselected defaults

Accessibility

- The W3C Web Accessibility Initiative provides introductions, solutions, and further resources to create accessible mobile websites and mobile web applications

Q2) Explain the effective use of screen real estate and features of mobile application

The first step to use the smaller interfaces of mobile devices effectively is to know the context of use. Who are the users, what do they need and why, and how, when, and where will they access and use information?

Mobile design is difficult, as developers try to elegantly display a telescoped view of almost limitless information. But user experience issues are amplified on mobile interfaces.

Cognitive load increases while attention is diverted by the needs to navigate, remember what was seen, and re-find original context.

Cognitive load is the mental effort to comprehend and use an application, whatever the inherent task complexity or information structure may be.

Effectively use screen real estate by embracing minimalism, maintaining a clear visual hierarchy, and staying focused.

Embrace Minimalism

• Limit the features available on each screen, and use small, targeted design features.

• Content on the screen can have a secondary use within an application, but the application designer should be able to explain why that feature is taking up screen space.

• Banners, graphics, and bars should all have a purpose.

Use a Visual Hierarchy

• Help users fight cognitive distractions with a clear information hierarchy.
• Draw attention to the most important content with visual emphasis.
• Users will be drawn to larger items, more intense colors, or elements that are called out with bullets or arrows; people tend to scan more quickly through lighter color contrast, less intense shades, smaller items, and text-heavy paragraphs.
• A consistent hierarchy means consistent usability; mobile application creators can create a hierarchy with position, form, size, shape, color, and contrast.

Stay Focused
• Start with a focused strategy, and keep making decisions to stay focused throughout development.
• A smaller file size is a good indicator of how fast an application will load, so the benefits of fighting feature creep extend beyond in-application user experience.
• Focused content means users won't leave because it takes too long for the overwhelming amount of images per screen to load.
• And users won't be frustrated with the number of links that must be cycled through to complete a task. Text-heavy pages reduce engagement as eyes glaze over and users switch to another application.
• If people have taken the time to install and open an application, there is a need these users hope to meet.
• Be methodical about cutting back to user necessities. Build just enough for what users need, and knowing what users need comes from understanding users
As it stands, there are really four major development targets. Each of the native frameworks comes with certain expectations and a user base. BlackBerry is often used in education and government, whereas the iPhone and Android user base is far more widespread. Windows Phone 7 being the newcomer is used primarily by developers and hasn't necessarily hit its stride yet.
iOS, the technology that is run on Apple mobile devices, has benefits and limitations specific to its development cycle. The base language is Objective-C, with Cocoa Touch as the interface layer. At this time iOS can be developed only using Apple's XCode, which can run only on a Macintosh.
The Android framework, on the other hand, is written in Java, and can be developed using any Java tools. The specific tooling recommended by Google and the Android community is Eclipse with the Android toolkit. Unlike iOS, it can be developed on PC, Mac, or Linux. Like Android, the BlackBerry device framework is also written in Java; however, it is limited in that the Emulator and Distribution tools run only on Windows at this time.
The newest native framework on the market is Windows Phone 7 and its framework sits on top of the Microsoft's .NET Framework. The language of choice is C# and the framework lies in a subset of Silverlight, Microsoft's multiplatform web technology. It also has the limitation that the Microsoft Windows Phone tools run only on Windows.

Q3) What are the components of android applications? Explain with example.

Activities — It is the presentation layer of application. The UI of your application is built around one or more extensions of the Activity class. Activities use Fragments and Views to layout and display information, and to respond to user actions. Compared to desktop development, Activities are equivalent to Forms.

 Services — These are the invisible workers of your application. Service components run without a UI, updating your data sources and Activities, triggering Notifications, and broadcasting Intents. They are used to perform long running tasks, or those that require no user interaction (such as network lookups or tasks that need to continue even when your application's Activities aren't active or visible.)

 Content Providers — Shareable persistent data storage. Content Providers manage and persist application data and typically interact with SQL databases. They are also the preferred means to share data across application boundaries. You can configure your application's Content Providers to allow access from other applications, and you can access the Content Providers exposed by others. Android devices include several native Content Providers that expose useful databases such as the media store and contacts.

 Intents — A powerful inter-application message-passing framework. Intents are used extensively throughout Android. You can use Intents to start and stop Activities and Services, to broadcast messages system-wide or to an explicit Activity, Service,

or Broadcast Receiver, or to request an action be performed on a particular piece of data.

 Broadcast Receivers — Intent listeners. Broadcast Receivers enable your application to listen for Intents that match the criteria you specify. Broadcast Receivers start your application to react to any received Intent, making them perfect for creating event-driven applications.

 Widgets — Visual application components that are typically added to the device home screen. A special variation of a Broadcast Receiver, widgets enable you to create dynamic, interactive application components for users to embed on their home screens.

 Notifications — Notifications enable you to alert users to application events without stealing focus or interrupting their current Activity. They're the preferred technique for getting a user's attention when your application is not visible or active, particularly from within a Service or Broadcast Receiver. For example, when a device receives a text message or an email, the messaging and Gmail applications use Notifications to alert you by flashing lights, playing sounds, displaying icons, and scrolling a text summary. You can trigger these notifications from your applications

4. What are the preliminary costs involved in mobile application development

Mobile app development costs related to hardware and software. The developer team needs devices to test the software on. And if you want to deploy your application to any public market, then your company will need accounts on the various markets (these often renew annually).

1.2.1 Hardware

To develop good mobile apps:

a. Need an intel based Mac:

 Can run Windows on them either virtually or on the bare metal.

 Expect to spend between $800 to $1600

b. Need multiple monitors

 When debugging any application requires 3 machines

 Emulator/simulator

 My Dev Tool (IDE)

 Web browser

 Having access to all of this information at once prevents context switching for a developer.

c. Examples of devices you can use to test the various platforms instead of emulator:

 Android 2.2 (Froyo): Motorola Droid 2

 Android 3.0 Tablet: Samsung Galaxy Tablet

 Apple iPod Touch: iPod Touch 3rd Generation

 Apple iPhone (versions 3.x and 4.x) (cell service): iPhone 3GS

 Apple iPhone (versions 4 and greater) (cell service): iPhone 4

 Apple iPad (WiFi or 3G for cell service testing): iPad 1

 Apple iPad (with camera): iPad 2 or iPad 3

 Windows Phone 7: Samsung Focus

1.2.2 Software

When developing mobile applications, there are few overlaps when it comes to software. To develop for iOS you need a Mac. To develop for BlackBerry you need Windows. For Java-based frameworks use Eclipse. Building HTML for PhoneGap can be done in your text editor of choice. Table 1.1 shows software needed for development.

Table 1.1 Software needed for development

Window Phone 7	Windows Phone SDK Visual Studio Express Expression Blend for Windows Phone
iOS	xCode 4, iOS SDK xCode 4.1, iOS SDK (on Mac OS X 107)
Android	Eclipse, Android SDK
BlackBerry	BlackBerry Eclipse, BlackBerry Plugin, BlackBerry Simulator (only works on Windows)
Titanium	Titanium Studio, Titanium Mobile SDK + Android software + iOS software
PhoneGap	PhoneGap Plugin + iOS software (Mac only) + Android software + Windows Phone 7 software (Windows only)
Any Framework Text Editors	Text Mate (Mac) Notepad++ (Windows)

1.2.3 Licenses and Developer Accounts

Table 1.2 shows the information regarding various accounts necessary to develop mobile app. One has to pay some amount for developer accounts per year.

Table 1.1 Software needed for development

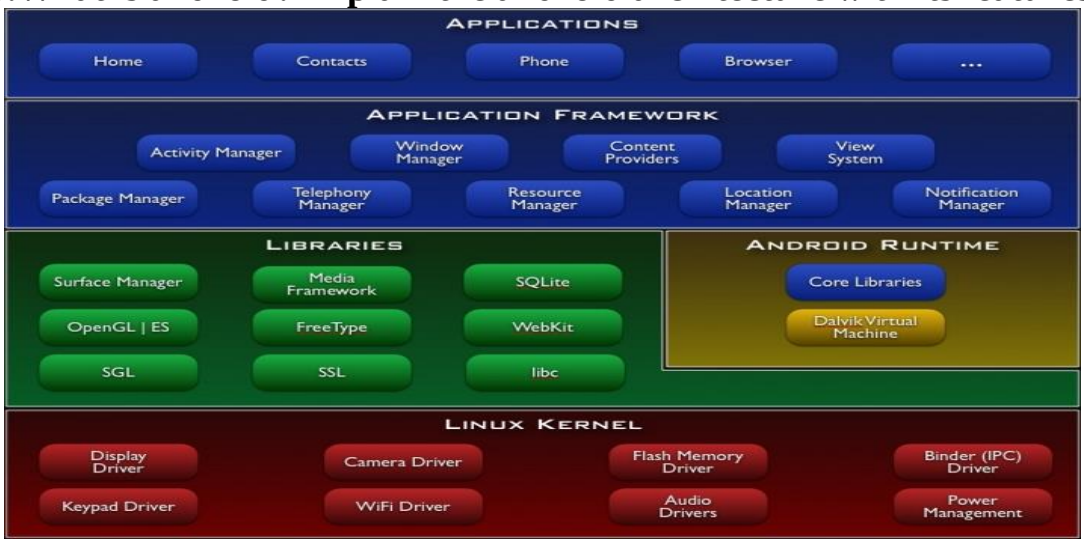
Platform	URL	Remarks
BlackBerry	http://us.blackberry.com/developers/ appworld/distribution.jsp	
Titanium	https://my.appcelerator.com/auth/ signup/offer/community	
Windows Dev Marketplace	http://create.msdn.com/ en-US/home/membership	Can submit unlimited paid apps, can submit only 100 free apps. Cut of Market Price to Store: 30%
Apple iOS Developer	http://developer.apple.com/ programs/start/standard/ create.php	Can only develop ad-hoc applications on up to 100 devices. Developers who publish their applications on the App Store will receive 70% of sales revenue, and will not have to pay any distribution costs for the application.
Android Developer	https://market.android.com/ publish/signup	Application developers receive 70% of the application price, with the remaining 30% distributed among carriers and payment processors.

1.2.4 Documentation and APIs

Following are links to the respective technologies' online documentation and APIs. This will be the location for the latest information in the respective technology

- **MSDN Library:** http://msdn.microsoft.com/enus/library/ff402535(v=vs.92).aspx
- **iOS Documentation:** http://developer.apple.com/devcenter/ios/index.action
- **BlackBerry Documentation:** http://docs.blackberry.com/en/developers/?userType=21
- **Android SDK Documentation:** http://developer.android.com/reference/packages.html and http://developer.android.com/guide/index.html
- **PhoneGap Documentation:** http://docs.phonegap.com/
- **Titanium API Documentation:** http://developer.appcelerator.com/apidoc/mobile/latest

5.What is android? Explain the android architecture with its features and diagram



The Android OS is roughly divided into five sections in four main layers:

Linux kernel — This is the kernel on which Android is based. This layer contains all the lowlevel device drivers for the various hardware components of an Android device.

Libraries — These contain all the code that provides the main features of an Android OS. For example, the SQLite library provides database support so that an application can use it for data storage. The WebKit library provides functionalities for web browsing.

Android runtime — At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the Dalvik virtual machine, which enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine (Android applications are compiled into the Dalvik executables). Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.

Application framework — Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications.

Applications — At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, etc.), as well as applications that you download and install from the Android Market. Any applications that you write are located at this layer.

6.Describe the basic views in android with a suitable code snippet.

Basic Views

Some of the basic views that can be used to design UI of Android application are:

- TextView
- EditText
- Button
- ImageButton
- CheckBox
- ToggleButton
- RadioButton
- RadioGroup

These basic views enable you to display text information, as well as perform some basic selection.

 TextView View: The TextView view is used to display text to the user. When we create a new Android project, Eclipse always creates one <TextView> element in activity_main.xml file, to display Hello World as shown below –

```
<?xml version=”1.0” encoding=”utf-8”?>
<LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”
android:orientation=”vertical”
android:layout_width=”fill_parent”
android:layout_height=”fill_parent”
>
<TextView
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”@string/hello”
/>
</LinearLayout>
```

Button — Represents a push-button widget

ImageButton — Similar to the Button view, but it also displays an image

EditText — A subclass of the TextView view, but it allows users to edit its text content

CheckBox — A special type of button that has two states: checked or unchecked

RadioGroup and **RadioButton** — The RadioButton has two states: either checked or unchecked. Once a RadioButton is checked, it cannot be unchecked. A RadioGroup is used to group together one or more RadioButton views, thereby allowing only one RadioButton to be checked within the RadioGroup.

ToggleButton — Displays checked/unchecked states using a light indicator

To understand the behavior of these views, create a new android project and place the following code in activity_main.xml file, without disturbing existing code.

```
<Button android:id=”@+id/btnSave”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Save” />
<Button android:id=”@+id/btnOpen”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”
android:text=”Open” />
<ImageButton android:id=”@+id/btnImg1”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:src=”@drawable/icon” />
<EditText android:id=”@+id/txtName”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content” />
<CheckBox android:id=”@+id/chkAutosave”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Autosave” />
<CheckBox android:id=”@+id/star”
style=”?android:attr/starStyle”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content” />
<RadioGroup android:id=”@+id/rdbGp1”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:orientation=”vertical” >
<RadioButton android:id=”@+id/rdb1”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Option 1” />
<RadioButton android:id=”@+id/rdb2”
```

```
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Option 2” />
</RadioGroup>
<ToggleButton android:id=”@+id/toggle1”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content” />
```

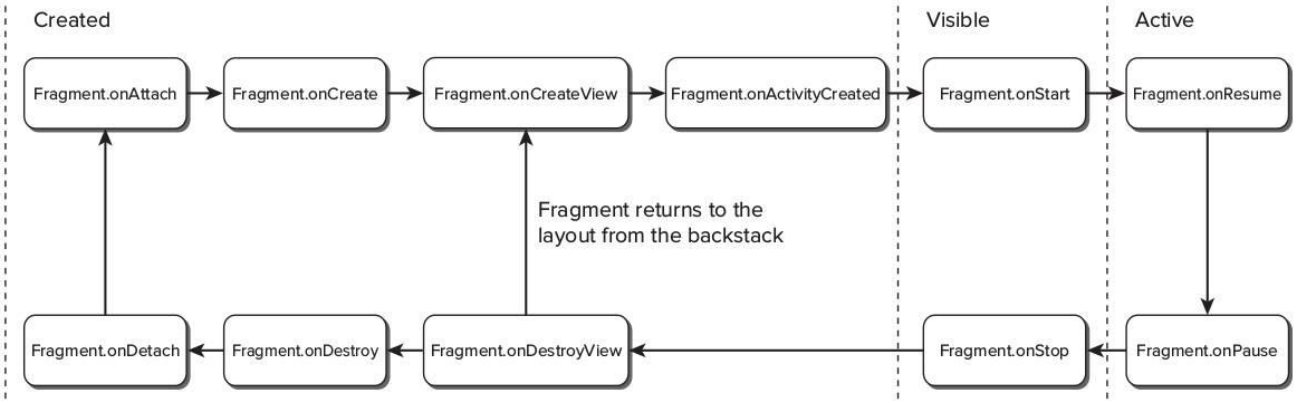
After running the application, the output will be displayed as shown in the following diagram. Click on each of these views, to observe their default behavior.

7. What is a fragment? Explain the lifecycle of a fragment

Fragments enable you to divide your Activities into fully encapsulated reusable components, each with its own lifecycle and UI. The primary advantage of Fragments is the ease with which you can create dynamic and flexible UI designs that can be adapted to various screen sizes.

Each Fragment is an independent module that is tightly bound to the Activity into which it is placed. Fragments can be reused within multiple activities, as well as laid out in a variety of combinations to suit multi-pane tablet UIs and added to, removed from, and exchanged within a running Activity to help build dynamic UIs. Fragments provide a way to present a consistent UI optimized for a wide variety of Android device types, screen sizes, and device densities.

The lifecycle events of a Fragment reflect those of its parent Activity. But, when the container Activity is in its active and resumed state by adding or removing a fragment, it will affect the lifecycle independently.



Attaching and Detaching Fragments from the Parent Activity: The full lifetime of your Fragment begins when it is bound to its parent Activity and ends when it has been detached. These events are represented by the calls to onAttach and onDetach, respectively. As with any handler called after a Fragment/Activity has become paused, it’s possible that onDetach will not be called if the parent Activity’s process is terminated without completing its full lifecycle. The onAttach event is triggered before the Fragment’s UI has been created, before the Fragment itself or its parent Activity have finished their initialization. Typically, the onAttach event is used to gain a reference to the parent Activity in preparation for further initialization tasks.

 Creating and Destroying Fragments: The created lifetime of your Fragment occurs between the first call to onCreate and the final call to onDestroy. As it’s not uncommon for an Activity’s process to be terminated without the corresponding onDestroy method being called, so a Fragment can’t rely on its onDestroy handler being triggered. As with Activities, you should use the onCreate method to initialize your Fragment. It’s good practice to create any class scoped objects here to ensure they’re created only once in the Fragment’s lifetime.

 Creating and Destroying User Interfaces: A Fragment’s UI is initialized (and destroyed) within a new set of event handlers: onCreateView and onDestroyView, respectively. Use the onCreateView method to initialize your Fragment:

o Inflate the UI,
o get references (and bind data to) the Views it contains,
o and then create any required Services and Timers.
Once you have inflated your View hierarchy, it should be returned from this handler:
return inflater.inflate(R.layout.my_fragment, container, false);

8. Explain the various information design tools of mobile interface design.

Sketching and Wireframes
o Sometimes we need to shape ideas on paper before focusing on the pixels.
o Storyboard application screens to outline features and flow, focusing on the big picture.
o Save wasted time developing the wrong thing the right way by involving all key stakeholders in the sketching and wire framing process.
o Mobile stencils are even on the market to help non doodlers pencil in ideas before turning to computer screens**.**
o A wireframe is a rough outline of each application's framework.
o Stay focused on functionality during wire framing;
 these easy-to-share,
 easy-to-edit files are just a skeleton of the design.
A simple image will do, but tools such as Balsamiq Mock-ups let designers drop boilerplate into a wireframe editor
 Mock-up Designs
o When you are ready to consider colors and fonts, you can build the mock-up design concept in Adobe Creative Suite.
o The final images of buttons and icons will be pulled from the final mock-up design, but details will solidify only after some experimentation.
o Look to existing stencils for a streamlined process that does not re-create the wheel.
Prototype:
o "Perfection is the enemy of good," and designs that start as ugly prototypes quickly progress to elegant, usable applications.
o The most primitive start is a most important iteration.
o Platform-specific tools are available, such as the Interface Builder or Xcode for iOS, but HTML and CSS are a standard and simple way to quickly build prototypical interactions
 On-device Testing:
o One of the most important tools during design will be the physical device.
o Buy, or Borrow, the devices and application will run on.
 Simulators and Emulators:
 Simulators and emulators are important when the hardware is unavailable and the service contracts for devices are prohibitively expensive.
 A simulator uses a different codebase to act like the intended hardware environment.
 An emulator uses a virtual machine to simulate the environment using the same codebase as the mobile application.
 It can be cost prohibitive to test on many devices, making emulators incredibly useful.
 Emulators can be run in collaboration with eye-tracking software already available in most testing labs, but an emulator lacks the touch experience of a mobile application.
 At an absolute minimum, use one of the target devices for user testing at this level.
During design, development, testing, and demonstration, these tools are incredibly valuable.

9.Briefly discuss the Gestalt's principles.

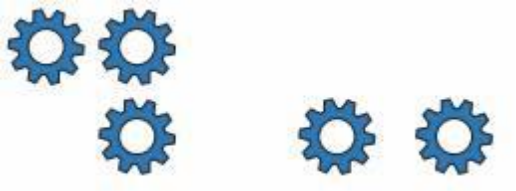
The ***Gestalt principles*** have had a considerable influence on design, describing how the human mind perceives and organizes visual data. The Gestalt principles refer to theories of visual perception developed by German psychologists in the 1920s. According to these

principles, every cognitive stimulus is perceived by users in its simplest form. Key principles include ***proximity, closure, continuity, figure and ground,*** and ***similarity.***

Proximity:

 Users tend to group objects together.

 Elements placed near each other are perceived in groups as shown in Figure



 Many smaller parts can form a unified whole.

 Icons that accomplish similar tasks may be categorically organized with proximity.

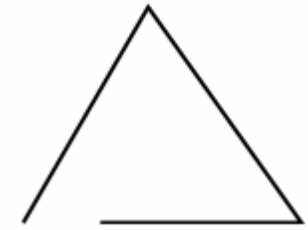
 Place descriptive text next to graphics so that the user can understand the relationship between these graphical and textual objects.

Closure:

 If enough of a shape is available, the missing pieces are completed by the human mind.

 In perceiving the unenclosed spaces, users complete a pattern by filling in missing information. For example, people recognize it as a triangle even though the Figure 1.2 is not complete.

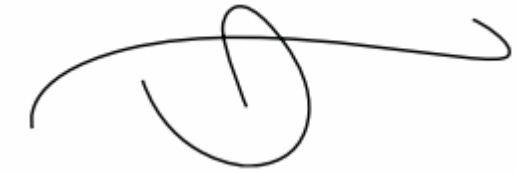
 In grid patterns with horizontal and vertical visual lines, use closure to precisely show the inside and outside of list items.



Continuity:

 The user's eye will follow a continuously-perceived object. When continuity occurs, users are compelled to follow one object to another because their focus will travel in the direction they are already looking.

 They perceive the horizontal stroke as distinct from the curled stroke in the Figure 1.3, even though these separate elements overlap.



Smooth visual transitions can lead users through a mobile application, such as a link with an indicator pointing toward the next object and task.

Figure and Ground:

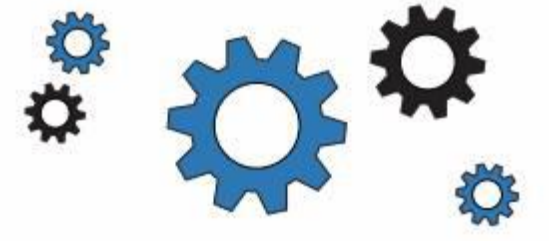
 A figure, such as a letter on a page, is surrounded by white space, or the ground. For example, in Figure 1.4, the figure is the gear icon, and the ground is the surrounding space.

Primary controls and main application content should maintain a distinct separation between figure and ground.

Similarity:

 Similar elements are grouped in a semi-automated manner, according to the strong visual perception of colour, form, size, and other attributes. Figure 1.5 illustrates it.

 In perceiving similarity, dissimilar objects become emphasized.
 Strict visual grids confuse users by linking unrelated items within the viewport.
 The layout should encourage the proper grouping of objects and ideas.



10. Describe the anatomy of android application.

The various folders and their files are as follows:
 src — Contains the file, ***MainActivity.java***. It is the source file for your activity. You will write the code for your application in this file.
 Android 4.4.2 — This item contains one file, ***android.jar***, which contains all the class libraries needed for an Android application.
 gen — Contains the ***R.java*** file, a compiler-generated file that references all the resources found in your project. ***You should not modify this file.***
 assets — This folder contains all the assets used by your application, such as HTML, text files, databases, etc.
 res — This folder contains all the resources used in your application. It also contains a few other subfolders:
o **drawable - <resolution>**: All the image files to be used by the Android application must be stored here.
layout - contains **activity_main.xml** file, which the is GUI of the application.
o **values -** contains files like **strings.xml, styles.xml** that are need for storing the string variables used in the applications, creating style-sheets etc.
 AndroidManifest.xml — This is the manifest file for your Android application. Here you specify the permissions needed by your application, as well as other features (such as intent-filters, receivers, etc.).
Details of some of the important files are given hereunder:
 strings.xml File: The activity_main.xml file defines the user interface for your activity. Observe the following in bold:
<TextView
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=**”@string/hello” />**
The ***@string*** in this case refers to the strings.xml file located in the res/values folder. Hence, ***@string/hello*** refers to the hello string defined in the ***strings.xml*** file, which is “Hello World!”:
<?xml version=”1.0” encoding=”utf-8”?>
<resources>
<string name=”hello”>Hello World!</string>
<string name=”app_name”>HelloWorld</string>
</resources>
It is recommended that you store all the string constants in your application in this ***strings.xml*** file and reference these strings using the ***@string*** identifier. That way, if you ever need to localize your application to another language, all you need to do is replace the strings stored in the ***strings.xml*** file with the targeted language and recompile your application.
 AndroidManifest.xml File: This file contains detailed information about the application. Observe the code in this file:
<?xml version=*"1.0"* encoding=*"utf-8"*?>
<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

```
package="com.example.HelloWorld"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk
android:minSdkVersion="19"
android:targetSdkVersion="19" />
<application
android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".MainActivity"
android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
</application>
</manifest>
```

Key points about this file are as below :

o It defines the package name of the application as ***net.learn2develop.HelloWorld***.

o The version code of the application is 1. This value is used to identify the version number of your application. It can be used to programmatically determine whether an application needs to be upgraded.

o The version name of the application is 1.0. This string value is mainly used for display to the user.

o The application uses the image named ***ic_launcher.png*** located in the ***drawable*** folder.

o The name of this application is the string named **app_name** defined in the ***strings.xml*** file.

o There is one activity in the application represented by the ***MainActivity.java*** file. The label displayed for this activity is the same as the application name.

o Within the definition for this activity, there is an element named <intent-filter>:

 The action for the intent filter is named ***android.intent.action.MAIN*** to indicate that this activity serves as the entry point for the application.

 The category for the intent-filter is named ***android.intent.category.LAUNCHER*** to indicate that the application can be launched from the device's Launcher icon.

o Finally, the ***android:minSdkVersion*** attribute of the <uses-sdk> element specifies the minimum version of the OS on which the application will run.

 R.java File: As you add more files and folders to your project, Eclipse will automatically generate the content of **R.java**, which at the moment contains the following:

```
package net.learn2develop.HelloWorld;
public final class R {
public static final class attr {
}
public static final class drawable {
public static final int icon=0x7f020000;
}
public static final class layout {
public static final int main=0x7f030000;
}
public static final class string {
```

```
public static final int app_name=0x7f040001;
public static final int hello=0x7f040000;
}
}
```

You are not supposed to modify the content of the R.java file; Eclipse automatically generates the content for you when you modify your project.

 MainActivity.java File: The code that connects the activity to the UI (activity_main.xml) is the setContentView() method, which is in the MainActivity.java file:

```
package net.learn2develop.HelloWorld;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity
{
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}
}
```

Here, **R.layout.main** refers to the **activity_main.xml** file located in the **res/layout** folder. As you add additional XML files to the **res/layout** folder, the filenames will automatically be generated in the **R.java** file. The **onCreate()** method is one of many methods that are fired when an activity is loaded.