CMR
INSTITUTE OF
TECHNOLOGY

USN

Internal Assessment Test 1 – Sep. 2020

Sub:	**Mobile Applications**							**Sub Code:**	**18MCA52**
Date:	**14/09/2020**	**Duration:**	**90 min's**	**Max Marks:**	**50**	**Sem & Sec:**	**V A & B**	**Branch:**	**MCA**

Note : Answer any full FIVE Questions

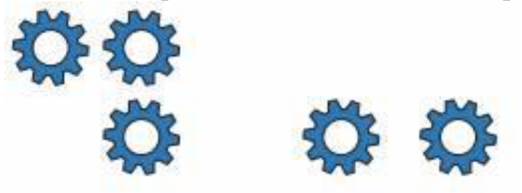
	PART I	**MARKS**	**OBE**	
			CO	**RBT**
1 a.	Briefly discuss the Gestalt's principles.	[5]	CO1	L1
b.	Design an application that accepts the URL and directs to it using intents	[5]	CO5	L3
2	Explain the effective use of screen real estate and features of mobile application	[10]	CO1	L2
3	What are the components of android applications? Explain with example.	[10]	CO2	L1,L2
4	What is android? Explain the android architecture with its features and diagram.	[10]	CO2	L2
5	Explain the various information design tools of mobile interface design.	[10]	CO1	L2
6	Design an application for login credentials which accepts user name and password from the user and display those details on the next screen.	[10]	CO5	L3
7	Describe the anatomy of android application.	[10]	CO2	L2
8	Design an application that accepts marks of different subjects from the user and display their name and grade on the other screen.	[10]	CO5	L3

1.a.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:

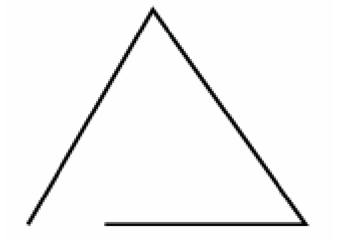
 Users tend to group objects together.
 Elements placed near each other are perceived in groups as shown in Figure



 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.

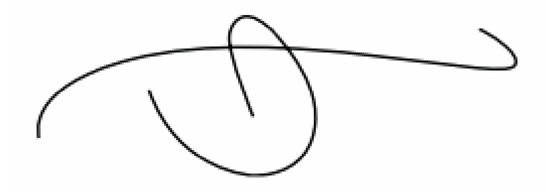
Closure:

 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.



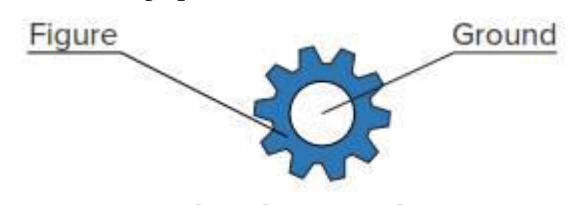
Continuity:

 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.



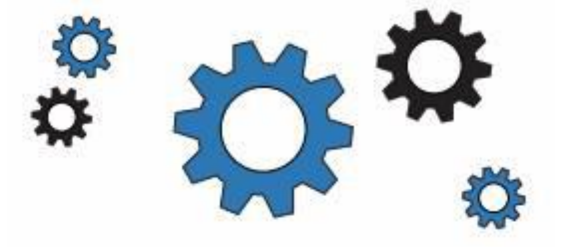
Smooth visual transitions can lead users through a mobile application, such as a link with an indicator pointing toward the next object and task.

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.



b) Design an application that accepts the URL and directs to it using intents

2) 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.

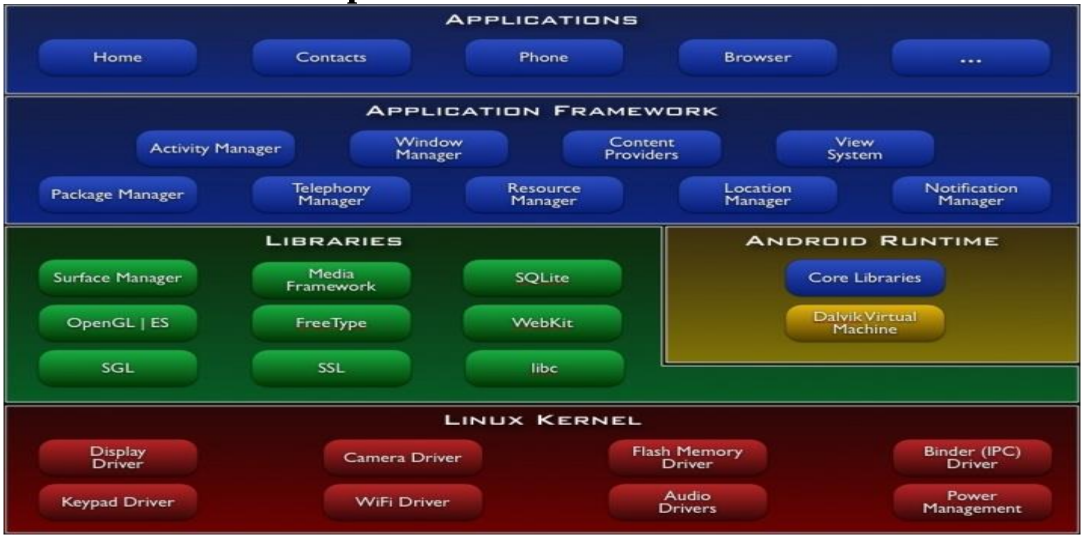
Stay Focused
• Start with a focused strategy, and keep making decisions to stay focused throughout development.
• A smaller file size is a good indicator of how fast an application will load, so the benefits of fighting feature creep extend beyond in-application user experience.
• Focused content means 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.

3) 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 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.

5. 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.

6. Design an application for login credentials which accepts user name and password from the user and display those details on the next screen.

7. 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.

8. Design an application that accepts marks of different subjects from the user and display their name and grade on the other screen.