CMR
INSTITUTE OF
TECHNOLOGY

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

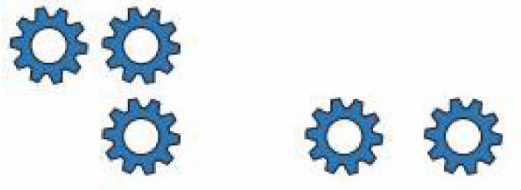
Internal Assessment Test 1 Answer Key– June. 2021									
Sub:	Mobile Application Development					Sub Code:	20MCA263	Branch:	MCA
Date:	24/06/2021	Duration:	90 min's	Max Marks:	50	Sem	IV		

Q1) Briefly discuss the Gestalt's principles (10 marks)

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:

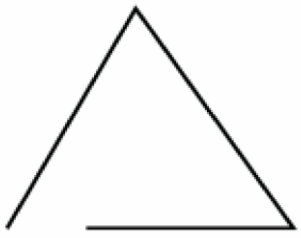
- Users tend to group objects together.
- Elements placed near each other are perceived in groups as shown in Figure



- 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.

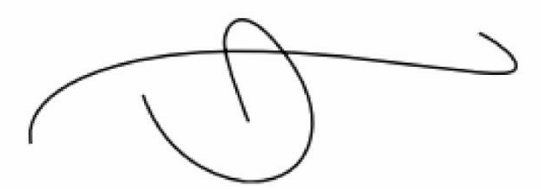
Closure:

- 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 below Figure 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:

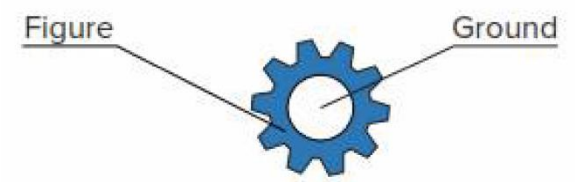
- The user's eye will follow a continuously-perceived object. When continuity occurs, users are compelled to follow one object to another because their focus will travel in the direction they are already looking.
- They perceive the horizontal stroke as distinct from the curled stroke in the below Figure, 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:

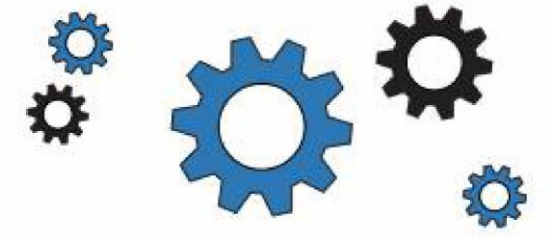
- A figure, such as a letter on a page, is surrounded by white space, or the ground.
- For example, in below Figure, the figure is the gear icon, and the ground is the surrounding space.



- 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.



Q2) What are the preliminary cost involved in mobile application development?

There are many costs associated with mobile application development.

- Each developer will need hardware and software to develop the applications on.
- The team will need 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).

Hardware

- To develop good mobile apps, you'll need an Intel-based Mac. Intel versions of Mac because you can run Windows on them either virtually (using something like Parallels, or VMWare Fusion) or on the bare metal (using Apple's BootCamp).
- You'll also need multiple monitors. The emulator/simulator running in one monitor, Dev Tool (IDE) running on another, and a web browser on another with the documentation for the platform for which you are developing. Having access to all of this information at once prevents context switching for a developer, and helps maintain focus.
- The emulator and simulators are great, but not perfect, so you'll need one of each of the types of devices you want to develop for.

Software

Following sections present an outline for what you will need for all of the platforms.

TABLE 1-1: Software Needed for Development

TARGETED FRAMEWORK	SOFTWARE REQUIRED
Window Phone 7	Windows Phone SDK Visual Studio Express Expression Blend for Windows Phone (Windows only)
iOS	xCode 4, iOS SDK xCode 4.1, iOS SDK (on Mac OS X 107) (Mac Only)
Android	Eclipse, Android SDK
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	TextMate (Mac) Notepad++ (Windows)

Licenses and Developer Accounts

The following table contains information regarding all of the various accounts necessary to develop for each platform and costs associated with such.

PLATFORM	URL	CAVEATS
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.

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/en-us/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

The Bottom Line

- Total cost per developer to create, maintain, and distribute mobile applications for all the platforms you can expect to pay a few thousand dollars just for the minimum infrastructure. And this is really the bare minimum for development.
- Given the opportunity to expand this more I would upgrade the laptop to a MacBook Pro, with plenty of RAM, and upgrade the hard disk drive (HDD) to a solid-state drive (SSD). By making these upgrades you will incur a higher initial cost, but the speed increase compared to the bare bones will recoup that cost, if only in peace of mind.
- It is difficult to quantify the savings from these upgrades, but developers without them are at a distinct disadvantage.

Q3) Describe the effective use of screen real estate

- 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

Q4) Describe the anatomy of android application

Various files that make up an Android project in the Package Explorer in Eclipse are as follows:

- src — Contains the .java source files for your project. In this example, there is one file, MainActivity.java. The MainActivity.java file is the source file for your activity. You will write the code for your application in this file.
- Android 2.3 library — 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: drawable-, layout, and values. Chapter 3 talks more about how you can support devices with different screen resolutions and densities.
- 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.).

The 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, MainActivity!":

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string name="hello">Hello World, MainActivity!</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.

The AndroidManifest.xml file contains detailed information about the application:

- It defines the package name of the application as net.learn2develop.HelloWorld.
- 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.
- The version name of the application is 1.0. This string value is mainly used for display to the user. You should use the format: .. for this value.
- The application uses the image named icon.png located in the drawable folder
- The name of this application is the string named app_name defined in the strings.xml file.
- 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.
- Within the definition for this activity, there is an element named
    - 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. Chapter 2 discusses intents in more details.
- Finally, the android:minSdkVersion attribute of the element specifies the minimum version of the OS on which the application will run. As you add more files and folders to your project, Eclipse will automatically generate the content of R.java. You are not supposed to modify the content of the R.javafi le; Eclipse automatically generates the content for you when you modify your project.

Finally, the code that connects the activity to the UI (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 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.

Q5) Develop android application to demonstrate life cycle of an activity

```
package net.learn2develop.Activities;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
```

```
public class MainActivity extends Activity {
 String tag = “Events”;
 /** Called when the activity is first created. */
 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
 Log.d(tag, “In the onCreate() event”)
}
 public void onStart()
 {
 super.onStart();
 Log.d(tag, “In the onStart() event”);
 }
 public void onRestart()
 {
 super.onRestart();
 Log.d(tag, “In the onRestart() event”);
 }
 public void onResume()
 {
 super.onResume();
 Log.d(tag, “In the onResume() event”);
 }
 public void onPause()
 {
 super.onPause();
 Log.d(tag, “In the onPause() event”);
 }
 public void onStop()
 {
 super.onStop();
 Log.d(tag, “In the onStop() event”);
 }
 public void onDestroy()
 {
 super.onDestroy();
 Log.d(tag, “In the onDestroy() event”);
 }
}
```

Q6a) Develop an android application to use an Intent object to bring up the Contacts application, from which the user can select the person to call

```
<?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” >
<Button android:id=”@+id/btn_chooseContact” android:layout_width=”fill_parent”
android:layout_height=”wrap_content” android:text=”Choose Contact” />
</ LinearLayout>
```

```
package net.learn2develop.Intents;
import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import android.net.Uri;
import android.provider.ContactsContract;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends Activity {
 Button  b4;
 int request_Code = 1;
 /** Called when the activity is first created. */
 @Override
 public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
b4 = (Button) findViewById(R.id.btn_chooseContact);
 b4.setOnClickListener(new OnClickListener()
 {
 public void onClick(View arg0){
 Intent i = new
 Intent(android.content.Intent.ACTION_PICK);
 i.setType(ContactsContract.Contacts.CONTENT_TYPE);
 startActivityForResult(i,request_Code);
 }
 });
 }
 public void onActivityResult(int requestCode, int resultCode, Intent data)
 {
 if (requestCode == request_Code)
 {
 if (resultCode == RESULT_OK)
 {
Toast.makeText(this,data.getData().toString(),
 Toast.LENGTH_SHORT).show();
 Intent i = new Intent(
 android.content.Intent.ACTION_VIEW,
 Uri.parse(data.getData().toString()));
 startActivity(i);
 }
}
}
}
```

Q6b) Explain resolving Intent Filter Collision with an appropriate example.

What happens if another activity (in either the same or a separate application) has the same filter name? For example, suppose your application has another activity named Activity3, with the following entry in the AndroidManifest.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Activities"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name" >
                  <! -- android:theme="@android:style/Theme.Dialog"  -->
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Activity2"
                  android:label="Activity 2">
            <intent-filter>
                <action android:name="net.learn2develop.ACTIVITY2" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity android:name=".Activity3"
                  android:label="Activity 3">
            <intent-filter>
                <action android:name="net.learn2develop.ACTIVITY2" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

If you call the startActivity() method with the following intent, then the Android OS will display a selection as shown in Figure 2-14: startActivity(new Intent("net.learn2develop.ACTIVITY2")); If you check the "Use by default for this action" item and then select an activity, then the next time the intent "net.learn2develop.ACTIVITY2" is called again, it will always launch the previous activity that you have selected. To clear away this default, go to the Settings application in Android and select Applications ➪ Manage applications and select the application name (see Figure 2-15). When the details of the application are shown, scroll down to the bottom and click the Clear defaults button.

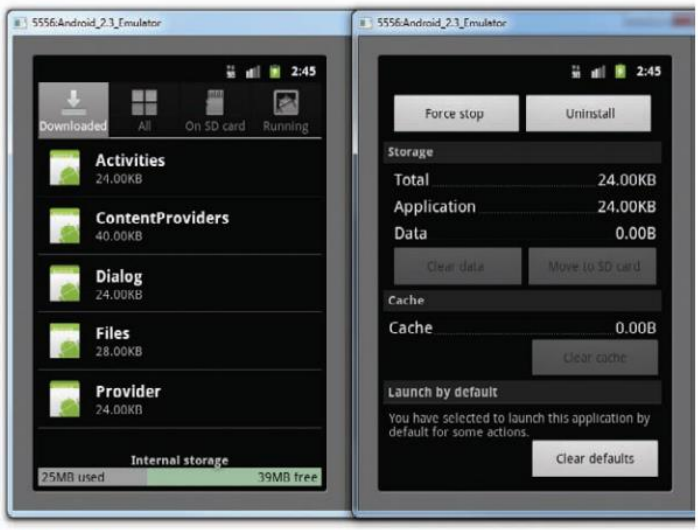
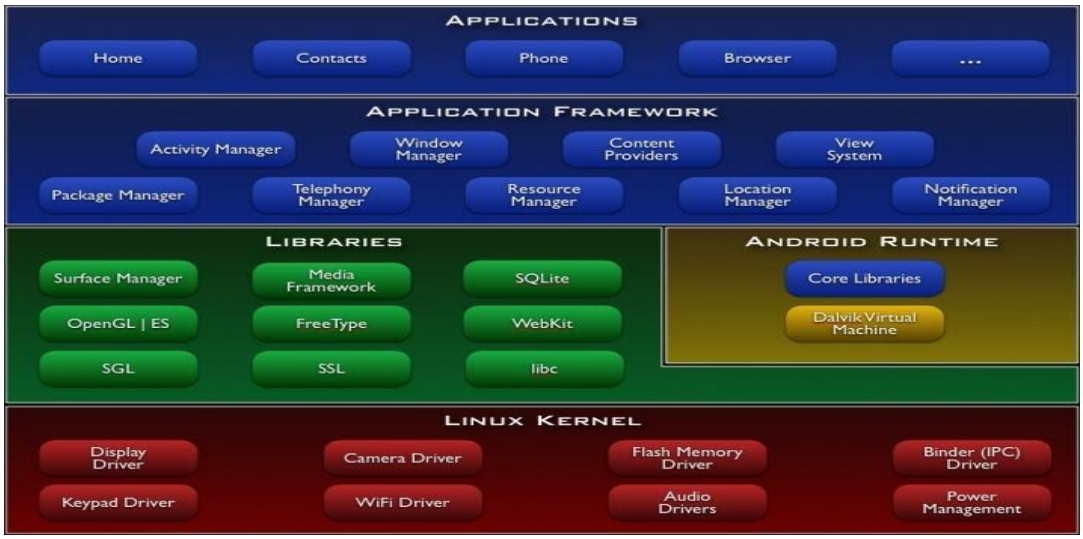


FIGURE 2-15

Q7) What is android? Explain the android architecture with its features and diagram.

- Android is a mobile operating system that is based on a modified version of Linux.
- It was originally developed by a start-up of the same name, Android, Inc.
- In 2005, as part of its strategy to enter the mobile space, Google purchased Android and took over its development work (as well as its development team).
- Android Inc. was founded in Palo Alto, California, in October 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White.
- Android is open and free; most of the Android code was 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.
- The main advantage of adopting Android is that it offers a unified approach to application development.
- Android OS is a Linux-based open source platform for mobile, cellular handsets developed by Google and the Open Handset Alliance
- The Open Handset Alliance (OHA) is a consortium of 84 firms to develop open standards for mobile devices.
- Member firms include HTC, Sony, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Google, Samsung Electronics, LG Electronics, T-Mobile, Sprint Corporation, Nvidia, and Wind River Systems.



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.

Q8) Explain the various information design tools of mobile interface design

Sketching and Wireframes

- Sometimes we need to shape ideas on paper before focusing on the pixels.
- Storyboard application screens to outline features and flow, focusing on the big picture.
- Save wasted time developing the wrong thing the right way by involving all key stakeholders in the sketching and wire framing process.
- Mobile stencils are even on the market to help non doodlers pencil in ideas before turning to computer screens**.**
- A wireframe is a rough outline of each application's framework.
- 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

- When you are ready to consider colors and fonts, you can build the mock-up design concept in Adobe Creative Suite.
- The final images of buttons and icons will be pulled from the final mock-up design, but details will solidify only after some experimentation.
- Look to existing stencils for a streamlined process that does not re-create the wheel.

Prototype:

- "Perfection is the enemy of good," and designs that start as ugly prototypes quickly progress to elegant, usable applications.
- The most primitive start is a most important iteration.
- 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:

- One of the most important tools during design will be the physical device.
- 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.

Q9) Develop an android application to accept user name and age, on a click of a button display on the Activity 2 if the user is eligible to vote. (10 marks)

Activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.bsec_intentpassdata.MainActivity" >



    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_marginLeft="22dp"
        android:layout_marginTop="23dp"
        android:text="Name" />

    <EditText
        android:id="@+id/ename"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="18dp"
        android:ems="10" >

        <requestFocus />
    </EditText>


    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="Age" />
<EditText
        android:id="@+id/eage"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:layout_marginLeft="18dp"
        android:ems="10" >

        <requestFocus />
    </EditText>



    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/textView1"
        android:layout_below="@+id/eage"
        android:layout_marginTop="26dp"
        android:text="Button" />

</RelativeLayout>
```

MainActivity.java

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;


public class MainActivity extends Activity {
      EditText ename, epass, eage ;
      Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

       btn=(Button) findViewById(R.id.button1);
       ename=(EditText) findViewById(R.id.ename);
       epass=(EditText) findViewById(R.id.epass);

       btn.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                  // TODO Auto-generated method stub
                  String uname=ename.getText().toString();
                  String age=eage.getText().toString();
                  int uage=Integer.parseInt(age);

                  //without bundle
                  Intent i = new Intent(MainActivity.this, Activity2.class);
                  i.putExtra("name", uname);
                  i.putExtra("age", uage);
                  startActivity(i);
            }
      });
    }
}
```

Activity2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:id="@+id/tname"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView1"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="47dp"
        android:text="TextView" />


</RelativeLayout>
```

Activity2.java

```
package com.example.bsec_intentpassdata;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Activity2 extends Activity{
      TextView tname, tage;

      @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity2);
        tname=(TextView) findViewById(R.id.tname);
        tage=(TextView) findViewById(R.id.tage);

        String msg="";
        int age=getIntent().getIntExtra("age",0);
        if(age>=18)
        {
            msg="Eligible to Vote";
        }
        else
            msg="Not Eligible to vote";

        tname.setText(getIntent().getStringExtra("name") + msg);



    }
}
```

Q10) Develop an android application to accept student usn and marks for 4 subjects, on a click of a button display the passing percentage of the student in the activity2. Each subject is of 100 marks.

Activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
```

```
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.bsec_intentpassdata.MainActivity" >



    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="23dp"
        android:text="USN" />

    <EditText
        android:id="@+id/eusn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="18dp"
        android:ems="10" >

        <requestFocus />
    </EditText>

        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="23dp"
        android:text="Marks1" />

    <EditText
        android:id="@+id/emarks1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="18dp"
        android:ems="10" >

        <requestFocus />
    </EditText>

      </EditText>

        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="23dp"
        android:text="Marks2" />

    <EditText
        android:id="@+id/emarks2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="18dp"
        android:ems="10" >

        <requestFocus />
    </EditText>

</EditText>

        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="23dp"
        android:text="Marks3" />

    <EditText
```

```
        android:id="@+id/emarks3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="18dp"
        android:ems="10" >

        <requestFocus />
    </EditText>
</EditText>

        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="23dp"
        android:text="Marks4" />

    <EditText
        android:id="@+id/emarks4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="18dp"
        android:ems="10" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="26dp"
        android:text="Button" />

</RelativeLayout>
```

MainActivity.java

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;


public class MainActivity extends Activity {
      EditText eusn, emark1, emark2, emark3, emark4;
      Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

       btn=(Button) findViewById(R.id.button1);
       eusn=(EditText) findViewById(R.id. eusn);
       emark1=(EditText) findViewById(R.id.emark1);
       emark2=(EditText) findViewById(R.id.emark2);
       emark3=(EditText) findViewById(R.id.emark3);
       emark4=(EditText) findViewById(R.id.emark4);

       btn.setOnClickListener(new OnClickListener() {

            @Override
```

```
            public void onClick(View v) {
                  // TODO Auto-generated method stub
                  String usn= eusn.getText().toString();
                  int mark1= Integer.parseInt (emark1.getText().toString());
                  int mark2= Integer.parseInt (emark2.getText().toString());
                  int mark3= Integer.parseInt (emark3.getText().toString());
                  int mark4= Integer.parseInt (emark4.getText().toString());

                  int percentage=(( mark1+ mark2+ mark3+ mark4)*100)/400;
                  Intent i = new Intent(MainActivity.this, Activity2.class);
                  i.putExtra("usn", usn);
                  i.putExtra("percentage", percentage);
                  startActivity(i);
            }
      });
    }
}
```

Activity2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:id="@+id/tname"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView1"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="47dp"
        android:text="TextView" />


</RelativeLayout>
```

Activity2.java

```
package com.example.bsec_intentpassdata;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Activity2 extends Activity{
      TextView tnam;

      @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity2);
        tname=(TextView) findViewById(R.id.tname);

        tname.setText(Integer.toString(getIntent().getIntExtra("percentage ",0)));




    }
}
```