CMR
INSTITUTE OF
TECHNOLOGY

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

Internal Assessment Test 1 – July. 2022

Sub:	Mobile Applications							Sub Code:	20MCA263
Date:	29/7/2022	Duration:	90 min's	Max Marks:	50	Sem:	II	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 Explain it.	[10]	CO1	L1
	OR			
2	Explain the terms to understand mobile application users.	[10]	CO1	L1
	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	Describe the anatomy of android application.	[10]	CO2	L2
	OR			
6	Compare and contrast mobile application platforms.	[10]	CO2	L2
	PART IV			
7	What is android? Explain the android architecture with diagram.	[10]	CO2	L2
	OR			
8	Explain the various information design tools of mobile interface design.	[10]	CO1	L2
	PART V			
9	Explain the effective use of screen real estate.	[10]	CO1	L2
	OR			
10	Write a mobile application code to accept and display course information like student name and course name.	[10]	CO2	L4

1. What do you understand by mobile information design Explain it.

- The visual display of information is how people connect with loved ones, colleagues, and friends.
- Information design includes
  - Information Display
  - Design Patterns
  - Content Structure and Usage

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.
- For example, a microwave has a simple display. When the timer alerts us the popcorn is done.
- 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. Ex: A microwave has a simple display.
- People identify signals
- interpret the meaning of these signals
- determine the goal accordingly

Design Patterns

A design pattern recycles and repurposes components, reusing the best ideas. More than time efficiency, patterns have been refined by use. Look to patterns, and maintain a pattern library that works for you, but 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.

1. Annunciator Panel

   An annunciator panel, seen at the top of Figure, gives information on the state of a mobile device. Though each mobile device will provide a slightly different panel, developers can modify or suppress the annunciator panel — which lists the hardware features such as network connection and battery power — within an application. Because the information in this area is

only notifications, application users will not usually have any direct interaction with the annunciator panel.

2. 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 revealable menu is already used on the same screen

➤ When a lack of controls, conflict with key interactions, or low discovery makes a revealable 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. Do not stack multiple fixed menus on top of each other, and reconsider the size and scope of an application if you have a fixed menu at both the top and bottom of a screen. The style, such as whether the menu goes across the top or bottom of the viewport, will be largely determined by the mobile device.

3. Expandable Menu

When all function options for one page cannot fit on the viewport, an expanding menu can provide selectively available options: a full menu similar to the one shown in Figure will reveal once prompted. A gesture, like a swipe or double tap, may prompt the reveal as well as selecting an on-screen icon. Soft keys — hardware buttons connected to on-screen labels — may also prompt a menu to reveal.

Users often benefit from multiple access options, especially for the harder-to-find gestural prompts. Users may exit a menu using a back button, a close button that is part of the revealed menu list, or by toggling with the same gesture or soft key that prompted the reveal. Try to keep close functionality on the screen while the menu is revealed.

4. Scroll
    - It is best to limit scrolling, limiting application screens to the size of the viewport whenever possible. Some types of apps you develop will require scrolling to present data to the user effectively, such as an email client. When scrolling must occur, checkthat the design communicates the area that can be scrolled, the current context of a user, and the ratio of the current view to the total content available
    - Only in-focus items should be able to scroll. Make an application more immersive, incorporating gestures such as tilting to scroll through content. Make sure that scrolling takes place only on a single axis, if possible. When scrolling must occur both horizontallyand vertically, consider providing a thumbnail of the user's place within the entire scrolling area.
    - The vertical list simply displays textual information, and is the foundation of informationdisplay on many mobile devices.
    - Graphical data — profile photos, category icons, status indicators — can clarify contentlists. Use position as well as size, contrast, and other identifiers to show the visual importance of elements that users utilize to sort information.
    - Thumbnails can be replaced by a default icon when a specific image is not available, butthe value of the graphical indicator diminishes as more icons are not individually identifiable. Use icons and images to emphasize clarity and categorical distinction — embracing a strong and varied silhouette — over personality or generic graphics.
    - An expandable list, reveals additional, valuable content for selected (touched) list itemswithout leaving the current view.
    - New information can be revealed as an animation whenever possible, aligning users with the structure and context of the expanded content areas. Higher-priority information, whether the revealed content or the list item title, should be set apart withsize, color, or contrast.
    - Users may scroll by gesture, device tilt, or on-screen buttons. Choose a different patternif live-scrolling — a pixel-by-pixel response to user input — is not possible.

5. Notifications and Feedback

   If the user needs new information, application creators must use notifications. These prompts pause ongoing tasks, and enable users to change processes and behaviors according to the new

information or feedback. Notification can inform a user (presenting a single button to close the notification), offer the ability to cancel a process or correct an error, or give multiple options from which to select. A user should never be presented with more than three options from any notification

Content and Controls

Input mechanisms, the controls users manipulate to enter or modify data, can be enhanced during the design process by keeping a few key concepts of mobile usability. Use standard controls, which positively shape user perceptions of an application.

Mobile platforms differ on the minimal size of any touch point; controls to navigate, select, and read content should be large enough for a finger to press without error. If users are most likely to hold a mobile device from the bottom with one hand, then frequently selected points in the interface are more usable when placed toward the bottom of the screen. Put these primary controls in reach of users' thumbs, and create wide-enough controls that left-handed users can use buttons and tabs as easily as right-handed users. Information at the top of the screen will be out of the immediate comfortable reach of the average thumb: display less commonly selected functionality at the top of the screen.

Be sure to minimize the need to enter data to access application features, especially text entry, which is often time-consuming and frustrating. Where data entry is actually necessary, consider the cross-platform differences in touch controls, and scale content for the various screen sizes and resolutions.

Reveal Content

You can display content either as a full page or revealed in context. When you use a full-page layout, the user should not need to constantly bounce back and forth between pages: all necessary information to use the features on that page should be displayed on that very page. If content might help users decide how to proceed, a quickly revealed, in-context design (such as a pop-up box) may be the better design choice

Display a range of information, from graphics to text, in an array such as an ordered list. Information can be contextually presented as an addition to other information, and be prioritized or sortable

A hierarchy of shows the parent-child relationship of information within the context of a larger information set. Large amounts of data can be structured in a way that is relevant and readable. Precise labels and distinct structure helps users to explore, opening and closing content to find their way around the presented information.

Expandable panes reveal a small amount of content without leaving the context of the current application screen. Though the expanding functionality will not work on some lower-end or older devices, the additional information or interactive elements can enhance user experience. However application information is displayed, lengthen the value of the application by replenishing content, adding levels or features, and building a community around the application.

Intervening Screens

Between delivering personalized functionality and life-changing brilliance to users, there will be times that content must load, or a device becomes locked while the user looks up from their device. During those times, be sure to include application branding with the title of the application, and do not display information from any previous screen.

The Home and Splash Screen

The home and splash screens show a default set of available information when the application starts, or after a task has been completed. Make it clear when users are on the homepage with a distinctly different screen. The first screen of an application is a great opportunity for branding, as well as a potential point of frustration for impatient users. Minimize the use of long, over-animated splash screens and slow-to-access main features.

One way to reduce wait time and increase user value is to store the last-opened screen, and display that screen the next time the application is opened. A useful design method to disguise slow launch times is to use the splash screen image as the application background: users will perceive a quicker entry to application features.

Parallax scrolling, where foreground and background content move at different speeds, will make the screen appear to have more depth. This technique helps users understand their current location within an application, and invite more immersive engagement.

Because frequent navigation to the home screen encourages users to jump between pages and between applications, avoid depending on the home screen for the continuous use of primary features. It is also preferable to send users to another actionable screen with some notification of the completed task instead of an idle screen: idle screens often send the message a task is complete, encouraging users to jump to another application.

The Loading Screen

An in-progress, “loading” screen signals when new data is loading, whether the user logs in to an account, enters search criteria, or is receiving an automated alert. Develop applications to load as quickly as possible to avoid showing the loading screen, but design to accommodate application limitations.

Include the estimated position in progress, and avoid showing information from the previous screen while newly requested data is loading. To shape user perceptions of load times and give the appearance of quicker data retrieval, place the progress indicator over a screen shot of the last screen. When there is sufficient delay, you can display advertising on the loading screen.

Advertising

Project stakeholders need to pay the bills, and advertising can certainly help. But application creators must find a balance of attention and integration to avoid two common mistakes: too-obvious advertising, and advertising that is confused for application content. Set apart advertisements with a strong border, a different color or distinct texture for the background, a full-width box, or — when the advertisement is smaller than the screen width — a different alignment than application content. Advertising styles and guidelines will vary across platforms, but advertisements must generally:

- Be clearly differentiated from application content.
- Remain unobtrusive to application features.
- Be actionable, and allow user interaction.
- Be legible.
- Use consistent styles and layout throughout application screens.

Advertisements may scroll within application content, or be locked to the viewport. Avoid animated advertisements, which distract from primary tasks and information. Whether content is paid advertising, curated by application administrators, or a help screen, the best content is useful content.

Content Structure and Usage

Mobile application users are there to consume, produce, and share content: it does not matter how pretty or useful the application may seem to stakeholders if content is worthless to users.

Users need to quickly locate and effectively use information. Page layouts must therefore reflect the mental models that users understand. Label key elements to make it clear where users are, in context of where they can go, as well as how to complete the current process. These content titles should always be horizontal, and set apart with a different background or surrounding box. Be sure to maintain consistent capitalization, using either sentence or title case throughout all headers. Titles can include icons, but these should be descriptive of the content, and not needlessly redundant or vague.

Structured, templated designs are valuable to great user experience: when people can predict which information will appear on what screen, users can more easily manipulate and navigate through mobile screens.

Information Architecture

Give every application a strong foundation by organizing, labeling, and identifying all content that users will perceive. The most common structure is the parent-child relationship of a hierarchy. A growing segment of interactive content is faceted, tagging chunks of content with attributes that can be filtered during regular application use.

When categorizing nonfaceted information, limit ambiguity with exclusive categories. It helps to take note of the most important category and keep all items in that menu the same theme. Be sure to

balance the breadth, the user's ability to scan the page according to viewport size, and depth of the architecture. Limit the scope to a depth of two to three levels down.

Wayfinding, or how users will orient themselves in a space and move around it, is generally managed by paths, nodes, edges, landmarks, and districts.

Consistent, simple navigation elements help users find and use the best information an application has to offer. Decide what information is necessary, because too many options can be dangerously frustrating to users. Mobile users consume information and complete simple, linear tasks; jumping between tasks and comparing information are still more common to desktop user experiences. But do not discount the future of mobile, and the possibility of complex information seeking and content production.

Typography

The central focus of every application will often be textual content. The fonts used in any design are far less important than the way traditional typography methods are used throughout a mobile application.

Size, shape, contrast, color, and position all matter. Type elements, and the relationship between type elements, should be immediately findable, consumable, and usable. Though sans serif fonts are generally considered easier to read on a viewport, serif or slab serif fonts have a proper place, such as bringing emphasis to a header.

Text layout and alignment should follow certain readability guidelines. In a left-to-right language, left alignment is preferred over justified or center alignment. Because of the thin screen space, bullet lists are easier to scan than tables, and single-column layouts will generally work best. Mobile interfaces compound the issues of web interfaces, and add new environmental factors and use cases.

Interactive and dynamic interfaces enable exciting new design capabilities, Such as animated notifications. But just because it's neat doesn't mean it is a good idea. If users only glance down at their screen when an animation is at a low-visibility point, they miss important information.

Plain Language

"Plain language" is the idea that content producers should speak in the language of their users, in a way that is clear and understandable to the audience. Content usability is one of the most important factors for task success, and plain language is usable language. Because of the limited space of a mobile device screen, these tips are especially relevant to mobile application creators:

➤ Omit unnecessary words. Take the first draft and cut it in half, and then see if you can cut it in half again (you probably can).

➤ Use the simplest form of a verb.

➤ Use short, simple words; avoid jargon and abbreviations.

➤ Use pronouns.

Mobile devices have tightly restricted widths; when determining readability, font size generally matters more than the number of characters per line. Precision is key to successful mobile content. Mobile plain language best practices also include:

➤ Focus keywords to the beginning or top of any screen.

➤ Use the same voice, preferably active voice, throughout the interface. Try to also use the same tense, when practical.

➤ If a product must be referenced, use a consistent product name.

➤ Correct unnecessarily mean or passive-aggressive error messages and task prompts.

Avoid redundant content. Plain language, and much of mobile design, is not about “dumbing down” the interface; it’s about elegant precision.

2. Explain the terms to understand mobile application users.

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 1.1.

Figure 1.1 Proximity

- 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.

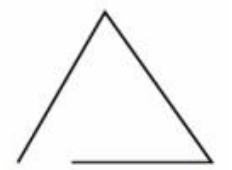


Figure 1.2 Closure

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.

Figure 1.3 Continuity

- 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.

Figure 1.4 Figure and Ground

- 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.

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

There are four different types of app components:

1. Activities
2. Services
3. Broadcast receivers
4. Content providers

Activities

An activity is the entry point for interacting with the user. It represents a single screen with a user interface. For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email app, each one is independent of the others. As such, a different app can start any one of these activities if the email app allows it. For example, a camera app can start the activity in the email app that composes new mail to allow the user to share a picture. An activity facilitates the following key interactions between system and app:

Keeping track of what the user currently cares about (what is on screen) to ensure that thesystem keeps running the process that is hosting the activity.

Knowing that previously used processes contain things the user may return to (stopped activities), and thus more highly prioritize keeping those processes around.

Helping the app handle having its process killed so the user can return to activities with their previous state restored. Providing a way for apps to implement user flows between each other,and for the system to coordinate these flows. (The most classic example here being share.)

You implement an activity as a subclass of the Activity class. For more information about the Activity class, see the Activities developer guide.

Services

A service is a general-purpose entry point for keeping an app running in the background for all kinds of reasons. It is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. For example, a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with anactivity. Another component, such as an activity, can start the service and let it run or bind to itin order to interact with it. There are actually two very distinct semantics services tell the system about how to manage an app: Started services tell the system to keep them running until their work is completed. This could be to sync some data in the background or play music even after the user leaves the app. Syncing data in the background or playing music also represent two different types of started services that modify how the system handles them:

Music playback is something the user is directly aware of, so the app tells the system this by saying it wants to be foreground with a notification to tell the user about it; in this case the system knows that it should try really hard to keep that service's process running, because the user will be unhappy if it goes away.

A regular background service is not something the user is directly aware as running, so the system has more freedom in managing its process. It may allow it to be killed (and then restarting the service sometime later) if it needs RAM for things that are of more immediate concern to the user.

Bound services run because some other app (or the system) has said that it wants to make use of the service. This is basically the service providing an API to another process. The system thus knows there is a dependency between these processes, so if process A is bound to a service in process B, it knows that it needs to keep process B (and its service) running for A. Further, if process A is something the user cares about, then it also knows to treat process B as something the user also cares about. Because of their flexibility (for better or worse), services have turned out to be a really useful building block for all kinds of higher-level system concepts. Live wallpapers, notification listeners, screen savers, input methods, accessibility services, and many other core system features are all built as services that applications implement and the system binds to when they should be running.

A service is implemented as a subclass of Service. For more information about the Service class, see the Services developer guide.

Broadcast receivers

A broadcast receiver is a component that enables the system to deliver events to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements. Because broadcast receivers are another well-defined entry into the app, the system can deliver broadcasts even to apps that aren't currently running. So, for example, an app can schedule an alarm to post a notification to tell the user about an upcoming event... andby delivering that alarm to a BroadcastReceiver of the app, there is no need for the app to remain running until the alarm goes off. Many broadcasts originate from the system—for example, a broadcast announcing that the screen has turned off, the battery is low, or a picturewas captured. Apps can also initiate broadcasts—for example, to let other apps know that some data has been downloaded to the device and is available for them to use. Although broadcast receivers don't display a user interface, they may create a status bar notification to alert the user when a broadcast event occurs. More commonly, though, a broadcast receiver is just a gateway to other components and is intended to do a very minimal amount of work. For

instance, it might schedule a JobService to perform some work based on the event with JobScheduler

A broadcast receiver is implemented as a subclass of BroadcastReceiver and each broadcast is delivered as an Intent object. For more information, see the BroadcastReceiver class.

Content providers

A content provider manages a shared set of app data that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access. Through the content provider, other apps can query or modify the data if the content provider allows it. For example, the Android system provides a content provider that manages the user's contact information. As such, any app with the proper permissions can query the content provider, such as ContactsContract.Data, to read and write information about a particular person

4 What are the preliminary costs involved in mobile application development

Preliminary cost associated with 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

1. 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).

2. 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.

3. 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.

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.

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.

5 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 theresources 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 storingthe 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 theapplication. Observe the code in this file:

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 theversion 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 fordisplay 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.

- 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 applicationcan 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**You are not supposed to modify the content of the R.java file; Eclipse automaticallygenerates 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 manymethods that are fired when an activity is loaded.

6. **Compare and contrast mobile application platforms.**

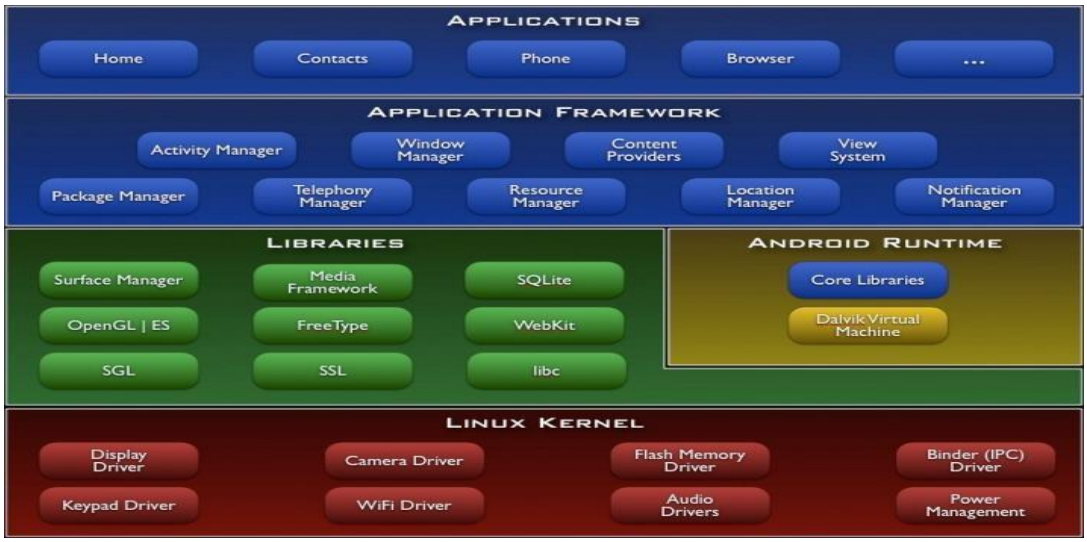
of different smartphones OS have been evaluated.

Table 2: Comparison of different mobile operating systems [15]-[19]

Parameters	Android	iOS	Symbian	Windows	Blackberry
OS Family	Linux	Darwin	RTOS	Window CE-7 Window NT-8	QNX
Vender	Open Handset Alliance, Google	Apple, Inc	Accenture on behalf of Nokia (historically Symbian Ltd. And Symbian Foundation)	Microsoft	Blackberry Ltd.
Environment (IDE)	Eclipse (Google)	XCode (Apple), Appcode	QT, Carbide.C++, Vistamax, Eclipse	Visual Studio	Eclipse, Blackberry JDE
CPU Architecture	ARM, x86, MIPS	ARM, ARM 64	ARM, x86	ARM	ARM
Source Model	Open Source and in most devices with proprietary components	Closed Source	Closed Source, Previously open source	Closed Source	Closed Source
License	Free and open source, but usually bundled with proprietary apps and drivers	Proprietary ULA except for open source components	Proprietary, Previously licensed under EPL	Proprietary	Proprietary
Written In	C, C++, Java	C, C++, Objective C, Swift	C, C++, ME, Python, Ruby, Flash Lite	C#, VB.NET, F#, C++, JScript	C, C++, HTML 5, Java script, CSS, Action script, Java
Market Share	48.8%	17.2%	0.1%	19.5%	11.1%
Market Size	Very High	High	Very low	Medium	Low
Application Store	Google Play	App Store	Nokia Ovi Store	Windows Phone Store	Blackberry World
Cross Platforming	Android supports cross platforming	iOS don't support cross platforming	Symbian supports cross plat forming	Windows support cross platforming	Blackberry don't support cross platforming
Memory Utilization	Paging, Memory Map, No Swapping	Automatic Reference Counting, No Garbage Collection	Memory management Unit and Cache resides on a SOC (System on Chip)	ROM/RAM is flash memory used for Virtual Memory storage. Programs can only run from main memory.	Contain slot for external memory supports 32GB Micro SD card at the time of full memory.
Security	Multi layer security. Most secure and usable OS	Low level software hardware and farm ware security	Gate keeper type of security every time ask for user permission	Windows OS does data encryption, leak prevention and	Blackberry provides two methods data encryption

7. What is android? Explain the android architecture with diagram.

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



ii)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.

iii) 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.

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

v)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.

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

Sketching and Wireframes

Sometimes we need to go analog, shaping 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 wireframing 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 wireframing; these easy-to-share, easy-to-edit files are just a skeleton of the design.

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.

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 an 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.

9. Explain the effective use of screen real estate.

The first step to use the smaller interfaces of mobile devices effectively is to know the contextof 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 almostlimitless information. But user experience issues are amplified on mobile interfaces.

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

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

Effectively use screen real estate by embracing minimalism, maintaining a clear visualhierarchy, 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 applicationdesigner 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 withbullets or arrows; people tend to scan more quickly through lighter color contrast, less intenseshades, smaller items, and text-heavy paragraphs.
- A consistent hierarchy means consistent usability; mobile application creators can create ahierarchy with position, form, size, shape, color, and contrast.

Stay Focused

- Start with a focused strategy, and keep making decisions to stay focused throughoutdevelopment.
- A smaller file size is a good indicator of how fast an application will load, so the benefits offighting feature creep extend beyond in-application user experience.
- Focused content means users won't leave because it takes too long for the overwhelmingamount 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 toanother application.

10. Write a mobile application code to accept and display course information like student name and course name.

```
<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.p1.MainActivity" >

<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginTop="26dp"
    android:text="StudentName" />

<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/textView1"
    android:layout_marginLeft="65dp"
    android:layout_toRightOf="@+id/textView1"
    android:ems="10" >

    <requestFocus />
</EditText>

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/textView1"
    android:layout_below="@+id/textView1"
    android:layout_marginTop="30dp"
    android:text="CourseName" />

<EditText
    android:id="@+id/editText2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/textView2"
    android:layout_alignBottom="@+id/textView2"
    android:layout_alignLeft="@+id/editText1"
    android:layout_marginLeft="20dp"
    android:ems="10" />

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/editText2"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="62dp"
    android:text="Submit" />
```

```
</RelativeLayout>
```

Java File

```
public class MainActivity extends ActionBarActivity {
EditText studentNameEdt;
EditText courseNameEdt;
Button loginBtn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // initializing our edit text  and buttons.
        studentNameEdt = (EditText) findViewById(R.id.editText1);
        courseNameEdt = (EditText) findViewById(R.id.editText2);
        loginBtn = (Button)findViewById(R.id.button1);

        // adding on click listener for our button.
        loginBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                // on below line we are getting data from our edit text.
                String studentName = studentNameEdt.getText().toString();
                String courseName = courseNameEdt.getText().toString();

                // checking if the entered text is empty or not.
               if (TextUtils.isEmpty(userName) &&
TextUtils.isEmpty(password)) {
                    Toast.makeText(MainActivity.this, "Please enter data",
Toast.LENGTH_SHORT).show();
                }

                else
                {
                  Toast.makeText(MainActivity.this, "Studentname
is"+studentName+"\t"+"courseName is"+courseName, Toast.LENGTH_SHORT).show();

                }
                // calling a method to login our user.



            }
        });
    }


}
```