CBCS SCHEME

USN 1CR19MCA76

18MCA52

Fifth Semester MCA Degree Examination, Jan./Feb. 2021
Mobile Applications

Time: 3 hrs. Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Module-1

1 a. What are the preliminary costs involved in mobile application development? Explain. (10 Marks)
b. Describe the effective use of screen real estate. (10 Marks)

OR

2 a. Explain the various information design tools of mobile interface design. (10 Marks)
b. Briefly discuss the Gestalt's principles. (10 Marks)

Module-2

3 a. List and explain the steps to create your first android project. (10 Marks)
b. What is Android? Explain the android architecture with its features and diagram. (10 Marks)

OR

4 a. What is an activity? Explain briefly the life cycle of an activity. (10 Marks)
b. Describe the anatomy of Android application. (10 Marks)

Module-3

5 a. Explain in detail how to design user interfaces using views and view groups. (10 Marks)
b. Discuss the basic views in Android with a suitable code snippet. (10 Marks)

OR

6 a. What are the different methods for getting location data? Explain. (10 Marks)
b. Discuss APK file deployment in detail and steps involved in publishing android application. (10 Marks)

Module-4

7 a. Explain the procedure for sending an SMS through android application with a code snippet. (10 Marks)
b. Define service. How do you create your own service in android? Explain with a snippet code. (10 Marks)

OR

8 a. What is the process for binding activities to services in android applications? (10 Marks)
b. Explain the concepts of networking in the terms of communicating between service and activity. (10 Marks)

Module-5

9 a. What is a program level in iOS? Discuss the various program levels available for an iOS developer. (10 Marks)
b. With a neat diagram, explain iOS project. (10 Marks)

OR

10 a. Describe the Anatomy of a Windows Phone-7 App. (10 Marks)
b. Write short note on other useful windows phone thing. (10 Marks)

* * * * *

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

1.2.1 Hardware

To develop good mobile apps:

a. Need an intel based Mac:
- Can run Windows on them either virtually or on the bare metal.
- Expect to spend between $800 to $1600

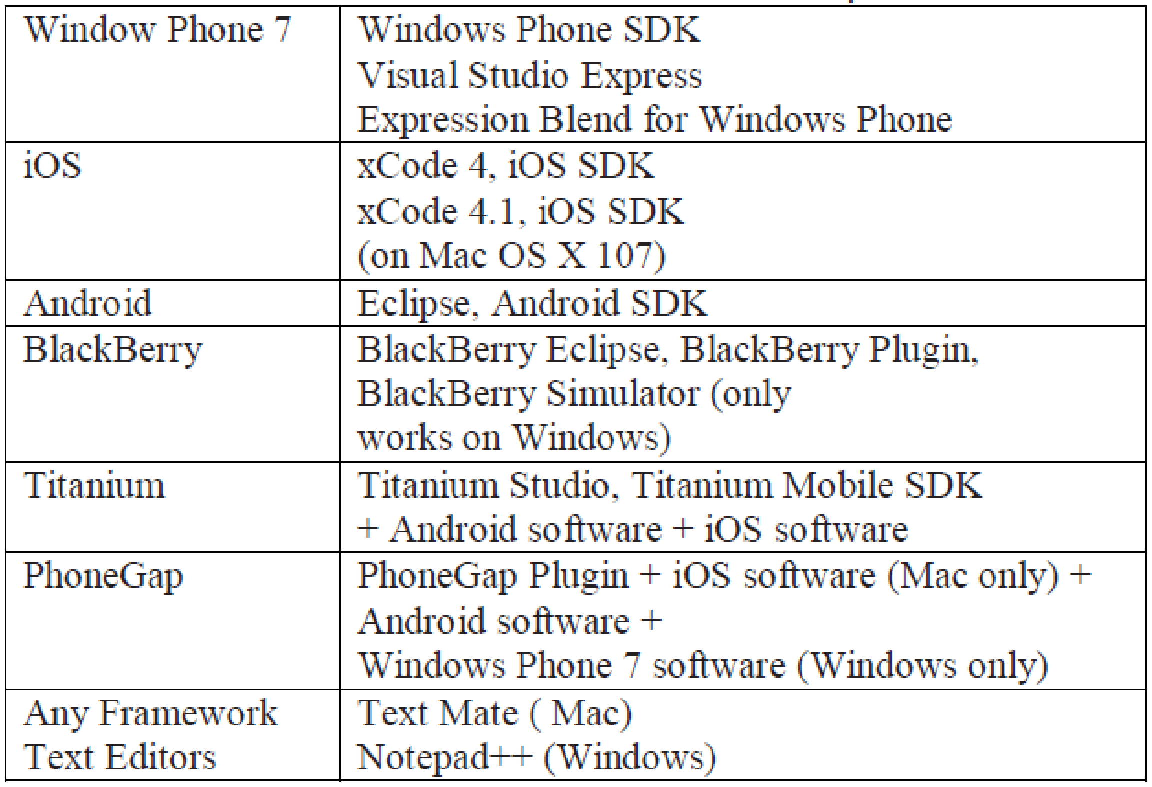
b. Need multiple monitors
- When debugging any application requires 3 machines
- Emulator/simulator
- My Dev Tool (IDE)
- Web browser
- Having access to all of this information at once prevents context switching for a developer.

c. Examples of devices you can use to test the various platforms instead of emulator:
- Android 2.2 (Froyo): Motorola Droid 2
- Android 3.0 Tablet: Samsung Galaxy Tablet
- Apple iPod Touch: iPod Touch 3rd Generation
- Apple iPhone (versions 3.x and 4.x) (cell service): iPhone 3GS
- Apple iPhone (versions 4 and greater) (cell service): iPhone 4
- Apple iPad (WiFi or 3G for cell service testing): iPad 1
- Apple iPad (with camera): iPad 2 or iPad 3
- Windows Phone 7: Samsung Focus

1.2.2 Software

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

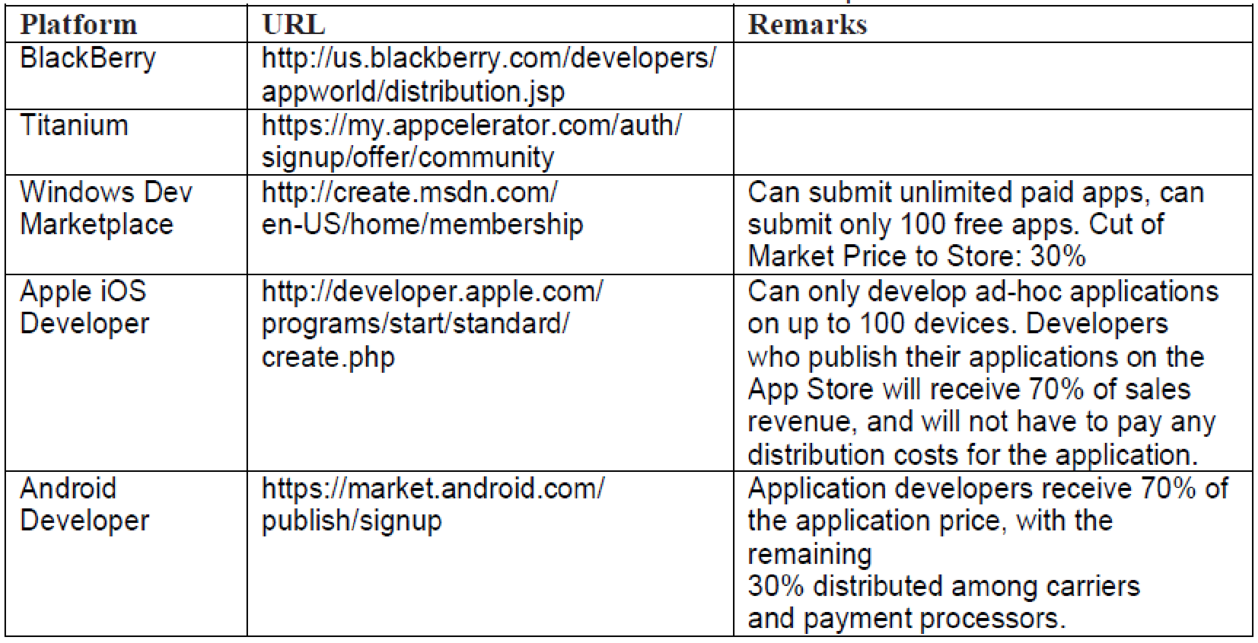
Table 1.1 Software needed for development



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

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

Table 1.1 Software needed for development



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

1.2.4 Documentation and APIs
Following are links to the respective technologies' online documentation and APIs. This will be the location for the latest information in the respective technology
 MSDN Library: http://msdn.microsoft.com/enus/library/ff402535(v=vs.92).aspx
 iOS Documentation: http://developer.apple.com/devcenter/ios/index.action
 BlackBerry Documentation: http://docs.blackberry.com/en/developers/?userType=21
 Android SDK Documentation: http://developer.android.com/reference/packages.html and http://developer.android.com/guide/index.html
 PhoneGap Documentation: http://docs.phonegap.com/
 Titanium API Documentation: http://developer.appcelerator.com/apidoc/mobile/latest

1.b. Explain 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

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.

2.a. Explain the various information design tools of mobile interface design.

Sketching and Wireframes

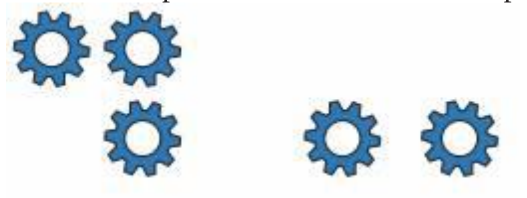
o Sometimes we need to shape ideas on paper before focusing on the pixels.
o Storyboard application screens to outline features and flow, focusing on the big picture.
o Save wasted time developing the wrong thing the right way by involving all key stakeholders in the sketching and wire framing process.
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

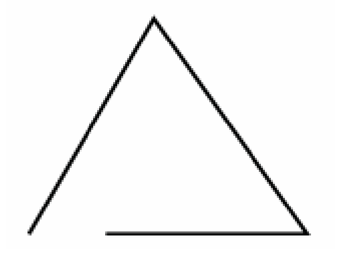
 Mock-up Designs

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.

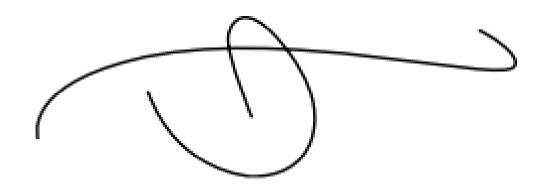
Prototype:
o "Perfection is the enemy of good," and designs that start as ugly prototypes quickly progress to elegant, usable applications.
o The most primitive start is a most important iteration.
o Platform-specific tools are available, such as the Interface Builder or Xcode for iOS, but HTML and CSS 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.

2.b. Briefly discuss the Gestalts 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

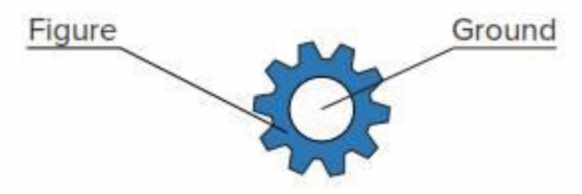
mind.
 In perceiving the unenclosed spaces, users complete a pattern by filling in missing information. For example, people recognize it as a triangle even though the Figure 1.2 is not complete.
 In grid patterns with horizontal and vertical visual lines, use closure to precisely show the inside and outside of list items.



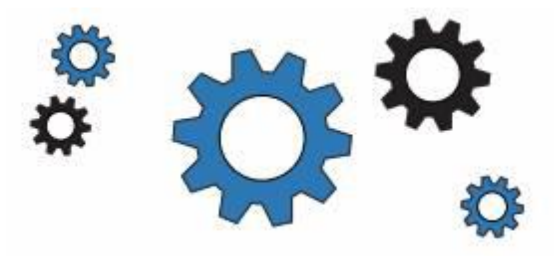
Continuity:
 The user’s eye will follow a continuously-perceived object. When continuity occurs, users are compelled to follow one object to another because their focus will travel in the direction they are already looking.
 They perceive the horizontal stroke as distinct from the curled stroke in the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

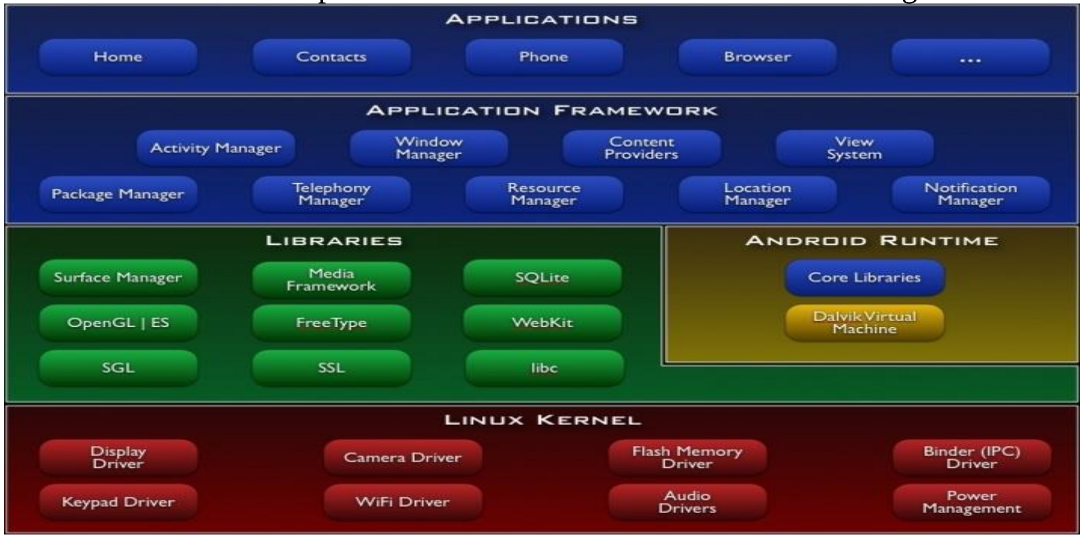


3.a. List and explain the steps to create your first android project.
Following are the steps involved in creating any Android application:
1. Using Eclipse, create a new project by selecting File ➪ New ➪ Android Application Project.
2. Name the Android project suitably, say *HelloWorld*.
3. In the Package Explorer (located on the left of the Eclipse IDE), expand the HelloWorld project by clicking on the various arrows displayed to the left of each item in the project. In the res/layout folder, double-click the activity_main.xml file. The activity_main.xml file defines the user interface (UI) of your application. The default view is the *Layout view*, which lays out the activity graphically. To modify the UI, click the activity_main.xml tab located at the bottom.
4. Add the following code in bold to the activity_main.xml file.

```
<LinearLayout
xmlns:android=http://schemas.android.com/apk/res/android
android:orientation=”vertical”
android:layout_width=”fill_parent”
android:layout_height=”fill_parent” >
<TextView
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”@string/hello” />
</LinearLayout>
```

5. To save the changes made to your project, press Ctrl+s.
6. You are now ready to test your application on the Android Emulator. Select the project name in Eclipse and press F11. You will be asked to select a way to debug the application. Select required Android Application and click OK.
7. The Android Emulator will now be started (if the emulator is locked, you need to slide the unlock button to unlock it first).
8. Click the Home button (the house icon in the lower-left corner above the keyboard) so that it now shows the Home screen.
9. Click the application Launcher icon to display the list of applications installed on the device. Note that the HelloWorld application is now installed in the application launcher.
(Note: In Step 6, include the steps of creating AVD).
In Android, an Activity is a window that contains the user interface of your applications. An application can have zero or more activities; in this example, the application contains one activity: MainActivity. This MainActivity is the entry point of the application, which is displayed when the application is started. When you debug the application on the Android

Emulator, the application is automatically installed on the emulator.
3.b. What is android? Explain the architecture with its features and diagrams.



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.

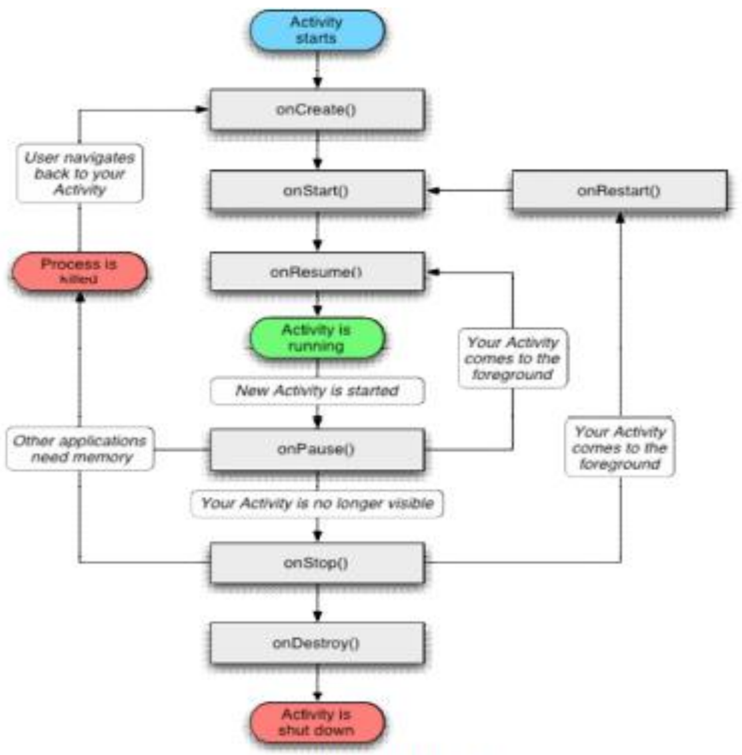
Libraries — These contain all the code that provides the main features of an Android OS. For example, the SQLite library provides database support so that an application can use it for data storage. The WebKit library provides functionalities for web browsing.

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.

4.a. What is an activity? Explain briefly the life cycle of an activity.
An activity is a window that contains the user interface of your applications. An application can have zero or more activities.


Activity starts
onCreate()
User navigates back to your Activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
Your Activity comes to the foreground
New Activity is started
Other applications need memory
onPause()
Your Activity comes to the foreground
Your Activity is no longer visible
onStop()
onDestroy()
Activity is shut down
Figure 2.2 Life Cycle of Activity

The Activity class defines the following events:

- onCreate() — Called when the activity is first created
- onStart() — Called when the activity becomes visible to the user
- onResume() — Called when the activity starts interacting with the user
- onPause() — Called when the current activity is being paused and the previous activity is being resumed
- onStop() — Called when the activity is no longer visible to the user
- onDestroy() — Called before the activity is destroyed by the system (either manually or by the system to conserve memory)
- onRestart() — Called when the activity has been stopped and is restarting again

4.b. 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.

5.a. Explain in detail how to design user interface using views and view groups.
Some of the basic views that can be used to design UI of Android application are:

 TextView
 EditText
 Button
 ImageButton
 CheckBox
 ToggleButton
 RadioButton
 RadioGroup

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

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

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

```
>
<TextView
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”@string/hello”
/>
</LinearLayout>
```

Button — Represents a push-button widget

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

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

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

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

 ToggleButton — Displays checked/unchecked states using a light indicator

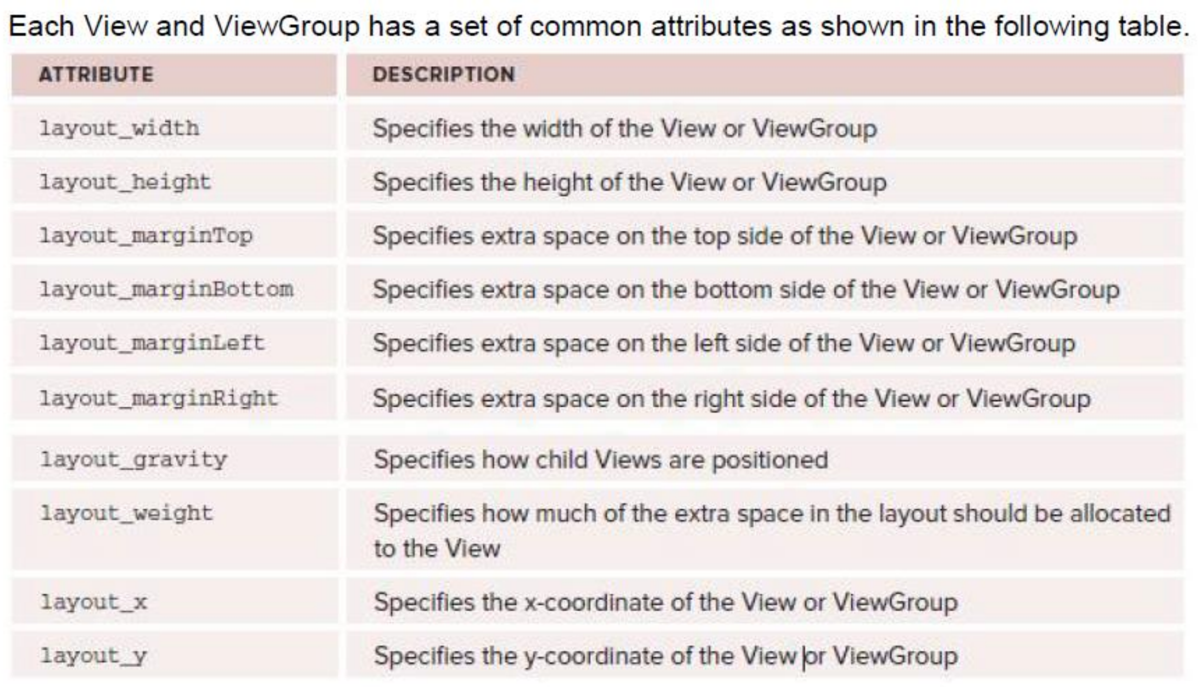
To understand the behavior of these views, create a new android project and place the following code in activity_main.xml file, without disturbing existing code.

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

```
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Option 1” />
<RadioButton android:id=”@+id/rdb2”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Option 2” />
</RadioGroup>
<ToggleButton android:id=”@+id/toggle1”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content” />
```
VIEW GROUPS

One or more views can be grouped together into a ViewGroup. A ViewGroup (which is itself a special type of view) provides the layout in which you can order the appearance and sequence of views. Examples of ViewGroups include LinearLayout and FrameLayout.

A ViewGroup derives from the base class android.view.ViewGroup.


Each View and ViewGroup has a set of common attributes as shown in the following table.

ATTRIBUTE	DESCRIPTION
layout_width	Specifies the width of the View or ViewGroup
layout_height	Specifies the height of the View or ViewGroup
layout_marginTop	Specifies extra space on the top side of the View or ViewGroup
layout_marginBottom	Specifies extra space on the bottom side of the View or ViewGroup
layout_marginLeft	Specifies extra space on the left side of the View or ViewGroup
layout_marginRight	Specifies extra space on the right side of the View or ViewGroup
layout_gravity	Specifies how child Views are positioned
layout_weight	Specifies how much of the extra space in the layout should be allocated to the View
layout_x	Specifies the x-coordinate of the View or ViewGroup
layout_y	Specifies the y-coordinate of the View or ViewGroup

Android supports the following ViewGroups:

- LinearLayout
- AbsoluteLayout
- TableLayout
- RelativeLayout
- FrameLayout
- ScrollView

5.b. Discuss the basic views in android with a suitable code snippet.

Basic Views

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

TextView

EditText
Button
ImageButton
CheckBox
ToggleButton
RadioButton
RadioGroup
These basic views enable you to display text information, as well as perform some basic selection.
 TextView View: The TextView view is used to display text to the user. When we create a new Android project, Eclipse always creates one <TextView> element in activity_main.xml file, to display Hello World as shown below –

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

Button — Represents a push-button widget
ImageButton — Similar to the Button view, but it also displays an image
EditText — A subclass of the TextView view, but it allows users to edit its text content
CheckBox — A special type of button that has two states: checked or unchecked
RadioGroup and RadioButton — The RadioButton has two states: either checked or unchecked. Once a RadioButton is checked, it cannot be unchecked. A RadioGroup is used to group together one or more RadioButton views, thereby allowing only one RadioButton to be checked within the RadioGroup.
ToggleButton — Displays checked/unchecked states using a light indicator
To understand the behavior of these views, create a new android project and place the following code in activity_main.xml file, without disturbing existing code.

```
<Button android:id=”@+id/btnSave”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Save” />
<Button android:id=”@+id/btnOpen”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content”
android:text=”Open” />
<ImageButton android:id=”@+id/btnImg1”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
```

```
android:src=”@drawable/icon” />
<EditText android:id=”@+id/txtName”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content” />
<CheckBox android:id=”@+id/chkAutosave”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Autosave” />
<CheckBox android:id=”@+id/star”
style=”?android:attr/starStyle”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content” />
<RadioGroup android:id=”@+id/rdbGp1”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:orientation=”vertical” >
<RadioButton android:id=”@+id/rdb1”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Option 1” />
<RadioButton android:id=”@+id/rdb2”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Option 2” />
</RadioGroup>
<ToggleButton android:id=”@+id/toggle1”
android:layout_width=”wrap_content”
android:layout_height=”wrap_content” />
```

After running the application, the output will be displayed as shown in the following diagram. Click on each of these views, to observe their default behavior.
6.a. What are the different methods for getting location data? Explain.
Nowadays, mobile devices are commonly equipped with GPS receivers.
o Because of the many satellites orbiting the earth, you can use a GPS receiver to find your location easily. However, GPS requires a clear sky to work and hence does not always work indoors or where satellites can’t penetrate (such as a tunnel through a mountain).
 Another effective way to locate your position is through *cell tower triangulation.*
o When a mobile phone is switched on, it is constantly in contact with base stations surrounding it.
o By knowing the identity of cell towers, it is possible to translate this information into a physical location through the use of various databases containing the cell towers’ identities and their exact geographical locations.
o The advantage of cell tower triangulation is that it works indoors, without the need to obtain information from satellites.
o It is not as precise as GPS because its accuracy depends on overlapping signal coverage, which varies quite a bit.

o Cell tower triangulation works best in densely populated areas where the cell towers are closely located.
 A third method of locating your position is to rely on Wi-Fi triangulation.
o Rather than connect to cell towers, the device connects to a Wi-Fi network and checks the service provider against databases to determine the location serviced by the provider.On the Android, the SDK provides the LocationManager class to help your device determine the user's physical location.
lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListener);
This method takes four parameters:
1. Provider -The name of the provider with which you register. In this case, you are using GPS to obtain your geographical location data.
2. minTime - The minimum time interval for notifications, in milliseconds.
3. minDistance - The minimum distance interval for notifications, in meters.
4. Listener - An object whose onLocationChanged() method will be called for each location update.

6.b. Discuss APK file deployment in detail and steps involved in publishing android application.
Once you have signed your APK fi les, you need a way to get them onto your users' devices.
Three methods are here:
 Deploying manually using the adb.exe tool
 Hosting the application on a web server
 Publishing through the Android Market
Besides the above methods, you can install your applications on users' devices through emails,SD card, etc. As long as you can transfer the APK file onto the user's device, you can install the application.
Using the adb.exe Tool: Once your Android application is signed, you can deploy it toemulators and devices using the adb.exe (Android Debug Bridge) tool (located in theplatform-tools folder of the Android SDK). Using the command prompt in Windows,navigate to the "<Android_SDK>\platform-tools" folder. To install the application to anemulator/device (assuming the emulator is currently up and running or a device is currentlyconnected), issue the following command:
adb install "C:\Users\Wei-Meng Lee\Desktop\LBS.apk"
(Note that, here, LBS is name of the project)
Besides using the adb.exe tool to install applications, you can also use it to remove an installed application. To do so, you can use the shell option to remove an application from its installed folder:
adb shell rm /data/app/net.learn2develop.LBS.apk
Another way to deploy an application is to use the DDMS tool in Eclipse. With an emulator (or device) selected, use the File Explorer in DDMS to go to the /data/app folder and use the "Push a file onto the device" button to copy the APK file onto the device.
Using a Web Server: If you wish to host your application on your own, you can use a web server to do that. This is ideal if you have your own web hosting services and want to provide the application free of charge to your users or you can restrict access to certain groups of people.
Following are the steps involved:
 Copy the signed LBS.apk fi le to c:\inetpub\wwwroot\. In addition, create a new HTML file named Install.html with the following content:

```
<html>
<title>Where Am I application</title>
<body>
Download the Where Am I application <a href=”LBS.apk”>here</a>
</body>
</html>
```

 On your web server, you may need to register a new MIME type for the APK file. The MIME type for the .apk extension is application/vnd.android.packagearchive.
 From the Application settings menu, check the “Unknown sources” item. You will be prompted with a warning message. Click OK. Checking this item will allow the Emulator/device to install applications from other non-Market sources (such as from a web server).
 To install the LBS.apk application from the IIS web server running on your computer, launch the Browser application on the Android Emulator/device and navigate to the URL pointing to the APK file. To refer to the computer running the emulator, you should use the special IP address of 10.0.2.2.
 Alternatively, you can also use the IP address of the host computer. Clicking the “here” link will download the APK file onto your device. Drag the notification bar down to reveal the download status. To install the downloaded application, simply tap on it and it will show the permission(s) required by this application.
 Click the Install button to proceed with the installation. When the application is installed, you can launch it by clicking the Open button.Besides using a web server, you can also e‑mail your application to users as an attachment; when the users receive the e‑mail they can download the attachment and
install the application directly onto their device.
Publishing on Android Market: It is always better to host your application on Android
market (Google Playstore). Steps involved in doing so, are explained hereunder:
 Creating a Developer Profile:
o Create a developer profile at http://market.android.com/publish/Home using a Google account.
o Pay one-time registration fees.
o Agree Android Market Developer Distribution Agreement
 Submitting Your Apps: If you intend to charge for your application, click the Setup Merchant Account link located at the bottom of the screen. Here you enter additional information such as bank account and tax ID. You will be asked to supply some details for your application.
Following are the compulsory details to be provided:
o The application in APK format
o At least two screenshots. You can use the DDMS perspective in Eclipse to capture screenshots of your application running on the Emulator or real device.A high-resolution application icon. This size of this image must be 512×512 pixels.
o Provide the title of your application, its description and recent update details.
o Indicate whether your application employs copy protection, and specify a content rating.
When all these setup is done, click Publish to publish your application on the Android Market.
7.a. Explain the procedure for sending an SMS through android application with a code snippet.
SMS messaging is one of the main *killer applications* on a mobile phone today — for some users as necessary as the phone itself. Any mobile phone you buy today should have at
least SMS messaging capabilities, and nearly all users of any age know how to send and
receive such messages. Android comes with a built-in SMS application that enables you to

send and receive SMS messages. However, in some cases you might want to integrate SMS capabilities into your own android application. For example, you might want to write an application that automatically sends a SMS message at regular time intervals. For example, this would be useful if you wanted to track the location of your kids — simply give them an Android device that sends out an SMS message containing its geographical location every 30 minutes.

4.1.1 Sending SMS Messages Programmatically

To create an application that can send SMS, following are the steps to be followed:

- Create a new android application.
- Add the following statements in to the main.xml file:

```
<Button
android:id=”@+id/btnSendSMS”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Send SMS” />
```

- In the AndroidManifest.xml file, add the following statements:

```
<uses-sdk android:minSdkVersion=”8” />
<uses-permission
android:name=”android.permission.SEND_SMS”>
</uses-permission>
```

- Add the following statements to the MainActivity.java file:

```
import android.app.PendingIntent;
import android.content.Intent;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity
{
Button btnSendSMS;
/** Called when the activity is first created.
@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
btnSendSMS = (Button) findViewById(R.id.btnSendSMS);
btnSendSMS.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
sendSMS(“5556”, “Hello my friends!”);
}
});
}
private void sendSMS(String phoneNumber, String message)
{
```

```
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(phoneNumber, null, message, null, null);
}
}
```

Following are the five arguments to the sendTextMessage() method:

- destinationAddress — Phone number of the recipient
- scAddress — Service center address; use null for default SMSC
- text — Content of the SMS message
- sentIntent — Pending intent to invoke when the message is sent
- deliveryIntent — Pending intent to invoke when the message has been Delivered

7.b. Define service. How do you create your own service in android? Explain with a snippet code.

A service is an application in Android that runs in the background without needing to interact with the user. For example, while using an application, you may want to play some background music at the same time. In this case, the code that is playing the background music has no need to interact with the user, and hence it can be run as a service. Services are also ideal for situations in which there is no need to present a UI to the user.

Following are the steps involved in creating own service.

- Create a new android application.
- Add a new class file to the project and name it MyService.java. Write the following code in it:

```
package net.learn2develop.Services;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;
public class MyService extends Service
{
@Override
public IBinder onBind(Intent arg0)
{
return null;
}
public int onStartCommand(Intent intent, int flags, int startId)
{
Toast.makeText(this,“ServiceStarted”,Toast.LENGTH_LONG).show();
return START_STICKY;
}
public void onDestroy()
{
super.onDestroy();
Toast.makeText(this,“ServiceDestroyed”,Toast.LENGTH_LONG).show();
}
}
```

The onBind() method enables you to bind an activity to a service. This in turn enables an activity to directly access members and methods inside a service. The onStartCommand() method is called when you start the service explicitly using the startService() method. This method signifies the start of the service, and you code it to do the things you need to do for your service. In this method, you returned the constant START_STICKY so that the service will continue to run until it is explicitly stopped. The onDestroy() method is called when the service is stopped using the stopService() method. This is where you clean up the resources used by your service.

 In the AndroidManifest.xml file, add the following statement :

```
<service android:name=”.MyService” />
```

 In the activity_main.xml file, add the following statements in bold:

```
<Button android:id=”@+id/btnStartService”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Start Service” />
<Button android:id=”@+id/btnStopService”
android:layout_width=”fill_parent”
android:layout_height=”wrap_content”
android:text=”Stop Service” />
```

 Add the following statements in bold to the MainActivity.java file:

```
import android.content.Intent;
import android.view.View;
import android.widget.Button;
public class MainActivity extends Activity
{
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
startService(new Intent(getBaseContext(), MyService.class));
}
});
Button btnStop = (Button) findViewById(R.id.btnStopService);
btnStop.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
stopService(new Intent(getBaseContext(), MyService.class));
}
});
}
}
```

8.a. What is the process for binding activities to services in android application?
Often a service simply executes in its own thread, independently of the activity that calls it. This doesn't pose any problem if you simply want the service to perform some tasks periodically and the activity does not need to be notified of the status of the service. For example, you may have a service that periodically logs the geographical location of the device to a database. In this case, there is no need for your service to interact with any activities, because its main purpose is to save the coordinates into a database. However, suppose you want to monitor for a particular location. When the service logs an address that is near the location you are monitoring, it might need to communicate that information
to the activity. In this case, you would need to devise a way for the service to interact with
the activity.

BINDING ACTIVITIES TO SERVICES
Real-world services are usually more sophisticated, requiring the passing of data so that they can do the job correctly for you. Using the service demonstrated earlier that downloads a set of files, suppose you now want to let the calling activity determine what files to download, instead of hardcoding them in the service.
First, in the calling activity, you create an Intent object, specifying the service name:

```
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
Intent intent = new Intent(getBaseContext(), MyService.class);
}
});
```

You then create an array of URL objects and assign it to the Intent object through its putExtra() method. Finally, you start the service using the Intent object:

```
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
Intent intent = new Intent(getBaseContext(), MyService.class);
try
{
URL[] urls = new URL[]
{
new URL("http://www.amazon.com/somefiles.pdf"),
new URL("http://www.wrox.com/somefiles.pdf"),
new URL("http://www.google.com/somefiles.pdf"),
new URL("http://www.learn2develop.net/somefiles.pdf")
};
intent.putExtra("URLs", urls);
} catch (MalformedURLException e)
{
e.printStackTrace();
}
startService(intent);
}
});
```
Note that the URL array is assigned to the Intent object as an Object array. On the service's end, you need to extract the data passed in through the Intent object in the onStartCommand() method:
```
@Override
public int onStartCommand(Intent intent, int flags, int startId)
{
// We want this service to continue running until it is explicitly
// stopped, so return sticky.
Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
Object[] objUrls = (Object[]) intent.getExtras().get("URLs");
URL[] urls = new URL[objUrls.length];
for (int i=0; i<objUrls.length-1; i++)
{
urls[i] = (URL) objUrls[i];
}
new DoBackgroundTask().execute(urls);
return START_STICKY;
}
```
The preceding first extracts the data using the getExtras() method to return a Bundle object. It then uses the get() method to extract out the URL array as an Object array. Because in Java you cannot directly cast an array from one type to another, you have to create a loop and cast each member of the array individually. Finally, you execute the background task by passing the URL array into the execute() method.This is one way in which your activity can pass values to the service. As you can see, if you have relatively complex data to pass to the service, you have to do some additional work to ensure that the data is passed correctly. A better way to pass data is to bind the activity directly to the service so that the activity can call any public members and methods on the service directly.

8.b. Explain the concepts of networking in the terms of communicating between service and activity.

One can communicate with the external world via SMS, email or using HTTP protocol. Using the HTTP protocol, you can perform a wide variety of tasks, such as downloading web pages from a web server, downloading binary data, and so on. The following project creates an Android project so that you can use the HTTP protocol to connect to the Web to download all sorts of data.

 Create a new android project and name it as Networking
 Add the following line in AndroidManifest.xml file:
```
<uses-permission
android:name="android.permission.INTERNET"></uses-permission>
```

 Import the following namespaces in the MainActivity.java file:

```
package net.learn2develop.Networking;
import android.app.Activity;
import android.os.Bundle;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.widget.ImageView;
import android.widget.Toast;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
```

 Define the OpenHttpConnection() method in the MainActivity.java file:

```
public class MainActivity extends Activity
{
private InputStream OpenHttpConnection(String urlString)
throws IOException
{
InputStream in = null;
int response = -1;
URL url = new URL(urlString);
URLConnection conn = url.openConnection();
if (!(conn instanceof HttpURLConnection))
throw new IOException(“Not an HTTP connection”);
try
{
HttpURLConnection httpConn = (HttpURLConnection) conn;
httpConn.setAllowUserInteraction(false);
httpConn.setInstanceFollowRedirects(true);
httpConn.setRequestMethod(“GET”);
httpConn.connect();
response = httpConn.getResponseCode();
if (response == HttpURLConnection.HTTP_OK)
{
in = httpConn.getInputStream();
}
}
catch (Exception ex)
{
throw new IOException("Error connecting");
}
return in;
}
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}
}
```

 Run the application to establish a connection.
 Working Procedure: Because you are using the HTTP protocol to connect to the Web, your application needs the INTERNET permission; hence, the first thing you do is add the permission in the AndroidManifest.xml file. You then define the OpenHttpConnection() method, which takes a URL string and returns an InputStream object. Using an InputStream object, you can download the data by reading bytes from the stream object. In this method, you made use of the HttpURLConnection object to open an HTTP connection with a remote URL. You set all the various properties of the connection, such as the request method, and so on. After you try to establish a connection with the server, you get the HTTP response code from it. If the connection is established (via the response code HTTP_OK), then you proceed to get an InputStream object from the connection. Using the InputStream object, you can then start to download the data from the server.

Communication between Services and Activities

Real-world services are usually more sophisticated, requiring the passing of data so that they can do the job correctly for you. Using the service demonstrated earlier that downloads a set of files, suppose you now want to let the calling activity determine what fi les to download, instead of hardcoding them in the service. Here is what you need to do. First, in the calling activity, you create an Intent object, specifying the service name:

```
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
Intent intent = new Intent(getBaseContext(), MyService.class);
}
});
```

You then create an array of URL objects and assign it to the Intent object through its putExtra() method. Finally, you start the service using the Intent object:

```
Button btnStart = (Button) findViewById(R.id.btnStartService);
btnStart.setOnClickListener(new View.OnClickListener()
{
public void onClick(View v)
{
Intent intent = new Intent(getBaseContext(), MyService.class);
try
{
URL[] urls = new URL[]
{
new URL(“http://www.amazon.com/somefiles.pdf”),
new URL(“http://www.wrox.com/somefiles.pdf”),
new URL(“http://www.google.com/somefiles.pdf”),
new URL(“http://www.learn2develop.net/somefiles.pdf”)
};
intent.putExtra(“URLs”, urls);
} catch (MalformedURLException e)
{
e.printStackTrace();
}
startService(intent);
}
});
```
Note that the URL array is assigned to the Intent object as an Object array. On the service’s end, you need to extract the data passed in through the Intent object in the onStartCommand() method:
```
@Override
public int onStartCommand(Intent intent, int flags, int startId)
{
// We want this service to continue running until it is explicitly
// stopped, so return sticky.
Toast.makeText(this, “Service Started”, Toast.LENGTH_LONG).show();
Object[] objUrls = (Object[]) intent.getExtras().get(“URLs”);
URL[] urls = new URL[objUrls.length];
for (int i=0; i<objUrls.length-1; i++)
{
urls[i] = (URL) objUrls[i];
}
new DoBackgroundTask().execute(urls);
return START_STICKY;
}
```
The preceding first extracts the data using the getExtras() method to return a Bundle object. It then uses the get() method to extract out the URL array as an Object array. Because in Java you cannot directly cast an array from one type to another, you have to create a loop and cast each member of the array individually. Finally, you execute the background task by passing the URL array into the execute() method.
This is one way in which your activity can pass values to the service. As you can see, if you have relatively complex data to pass to the service, you have to do some additional work to

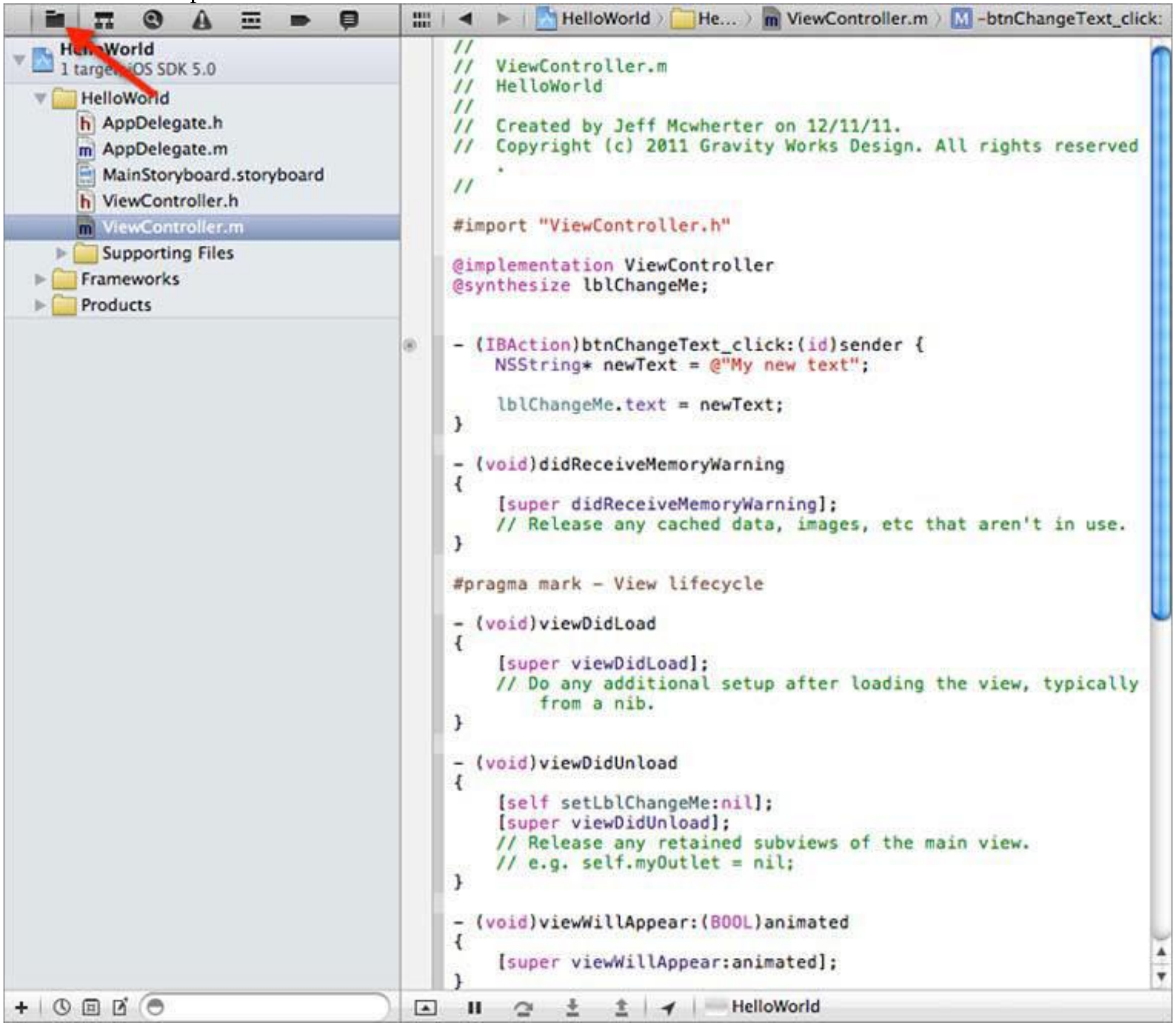
ensure that the data is passed correctly. A better way to pass data is to bind the activity directly to the service so that the activity can call any public members and methods on the service directly.

9.a. What is a program level in IOS? Discuss the various program levels available for an IOS developer.
The program level is where development teams, stakeholders, and other resources are devoted to some important, ongoing system development mission. It describes the program level teams, roles, and activities that incrementally deliver a continuous flow of value.
iOS Developer Program
This program level allows developers to distribute apps in the App Store as an individual, a sole proprietor, a company, an organization, a government entity, or an educational institution. The cost for this program is $99 a year.
iOS Developer Enterprise Program
This program level allows developers to develop proprietary apps for internal distribution within your company, organization, government entity, or educational institution. The cost for this program is $299 a year.
iOS Developer University Program
This program level allows higher-education institutions to create teams of up to 200 developers that can develop iOS applications. This program level is free, and allows for programs to be tested on physical devices, but does not allow for ad hoc or App Store deployment.

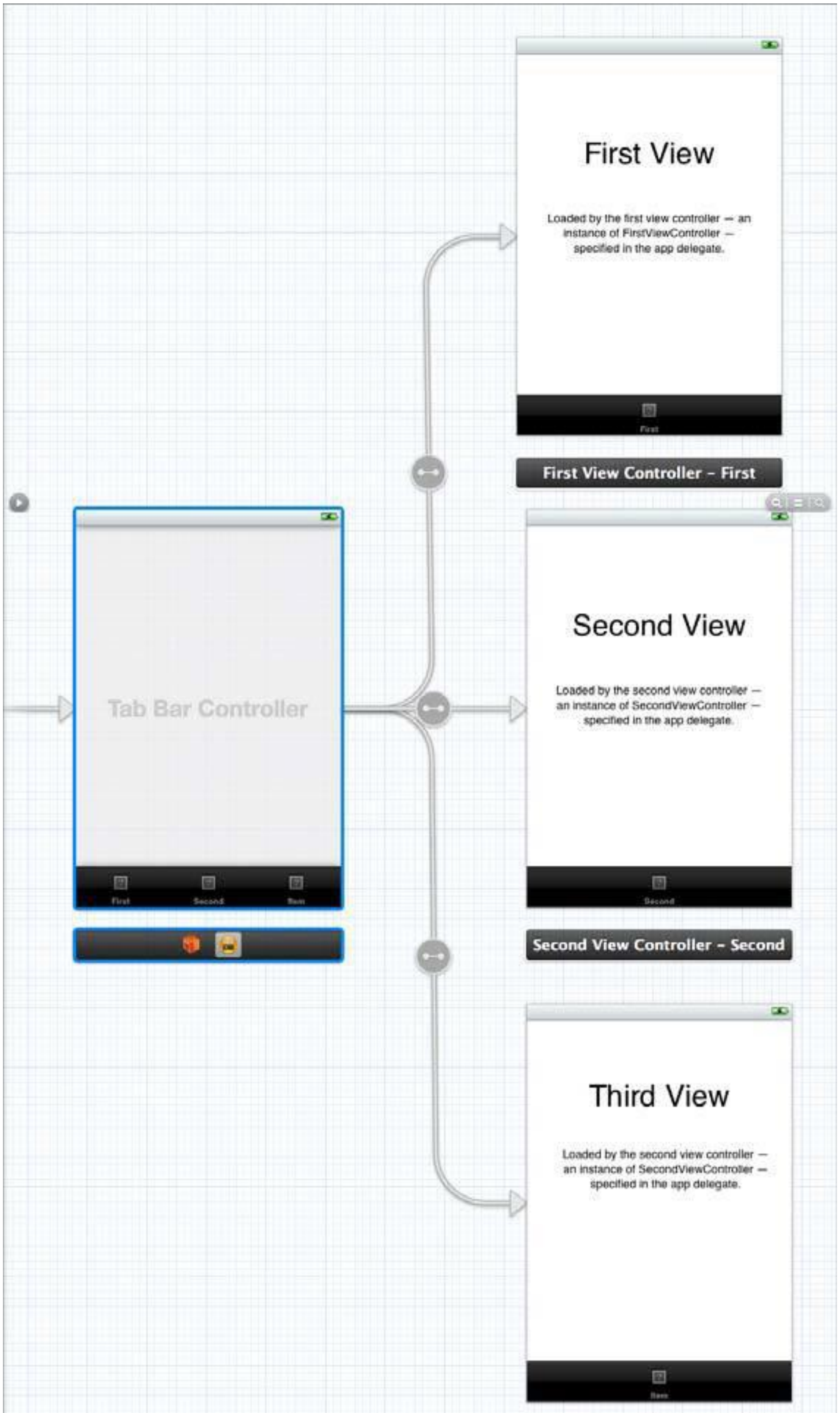
9.b. With a neat diagram, explain IOS project.
There are many steps to be followed before you start creating an IOS application. Here we will discuss few of them.
Anatomy of an iOS App
The files that are actually deployed to the iOS device are known as .app files and these are just a set of directories. Although there is an actual binary for the iOS application, you can open the .app
file and find the images, meta data, and any other resources that are included.
 Views: iPhone apps are made up of one or more views. Views usually have GUI elements such as text fields, labels, buttons, and so on. You can build a view built using the Interface Builder tool, which enables you to drag and drop controls on the view, or you can create a view entirely with code.
 Code that makes the Views work: Because iOS applications follow the MVC design pattern, there is a clean break between the UI and code that provides the application code.
 Resources: Every iOS application contains an icon file, an info.plist file that holds information about the application itself and the binary executable. Other resources such as images, sounds, and video are also classified as resources.
 Project Structure in Depth: When an iOS project is created within xCode, the IDE creates a set of files that are ready to run. These files provide the basics of what is needed to get going with a new project.
o Main.m: As with any C program, the execution of Objective-C applications start from the main() function, which is the main.m file.
o AppDelegate.m: The AppDelegate receives messages from the application object during the lifetime of your application. The AppDelegate is called from the

operating system, and contains events such as the
didFinishLaunchingWithOptions, which is an event that iOS would be
interested in knowing about.
o MainStoryBoard.storyboard: This is where the user interface is created. In past
versions of xCode/iOS the user interface was stored within .xib (pronounced NIB)
files. Although this method is still supported, Storyboards are a great improvement
over .xib files for applications with complex navigation and many views.
o Supporting Files: The supporting files directory contains files such as the plist
setting files (which contain customizable application settings), as well as string
resource files that are used within your app.
 Getting to Know the xCode IDE
It is important to use the correct tool for the job, regardless of whether you are constructing a
house
or constructing an application. If you are new to xCode, there will be a bit of a learning curve to
becoming proficient with the IDE, but xCode is a top-notch IDE with many features for you to
discover.
 Navigators: The left side of the xCode window is known as the navigator area. A variety of
navigators enable you to list the contents of your project, find errors, search for code, and
more. The remainder of this section introduces the Project Navigator, the Search Navigator,
and the Issue Navigator. Going from left to right, the project navigator is the first of the
xCode navigators; the icon looks like a file folder. The Project Navigator simply shows the
contents of your project or workspace, as shown in Figure 5.1. Double-clicking a file in the
Project Navigator opens the file in a new window, and single-clicking opens the file within

the xCode workspace.



```
//
//  ViewController.m
//  HelloWorld
//
//  Created by Jeff Mcwherter on 12/11/11.
//  Copyright (c) 2011 Gravity Works Design. All rights reserved
    .
//

#import "ViewController.h"

@implementation ViewController
@synthesize lblChangeMe;


- (IBAction)btnChangeText_click:(id)sender {
    NSString* newText = @"My new text";

    lblChangeMe.text = newText;
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Release any cached data, images, etc that aren't in use.
}

#pragma mark - View lifecycle

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically
        from a nib.
}

- (void)viewDidUnload
{
    [self setLblChangeMe:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
}
```



Loaded by the first view controller — an instance of FirstViewController — specified in the app delegate.

Loaded by the second view controller — an instance of SecondViewController — specified in the app delegate.

Loaded by the second view controller — an instance of SecondViewController — specified in the app delegate.

Storyboards: In iOS versions prior to iOS 5, developers needed to create a separate XIB file for each view of their application. A XIB file is an XML representation of your controls and instance variables that get compiled into the application. Managing an application that contains more than a few views could get cumbersome. iOS 5 contained a new feature called storyboards that enables developers to lay out their workflow using design tools built within xCode. Apps that use navigation and tab bars to transition between views are now much easier to manage, with a visual representation of how the app will flow. With Storyboards, you will have a better conceptual overview of all the views in your app and the connections between them. Figure shows an example of a storyboard for an application containing a tab bar for navigation to three other views.
10.a. Describe the anatomy of a Windows Phone-7 App.
Here we discuss the basic design elements used in Windows Phone 7 application development, and how you can leverage the tools you have at hand to implement them.

1. Storyboards: Storyboards are Silverlight's control type for managing animations in code. They are defined in a given page's XAML and leveraged using code behind. Uses for these animations are limited only by the transform operations you are allowed to perform on objects.

Anytime you want to provide the user with a custom transition between your pages or element updates, you should consider creating an animation to smooth the user experience.
Because storyboards are held in XAML you can either edit them manually or use Expression Blend's WYSIWYG editor.

In Blend, in the Objects and Timelines Pane at the left, click the (+) icon to create a storyboard. Once you have a storyboard, you can add key frames on your time line for each individual element you would like to perform a transformation on. This can include moving objects and changing properties (like color or opacity). After setting up your time line, you can start the storyboard in code. The name you created for your storyboard will be accessible in code behin

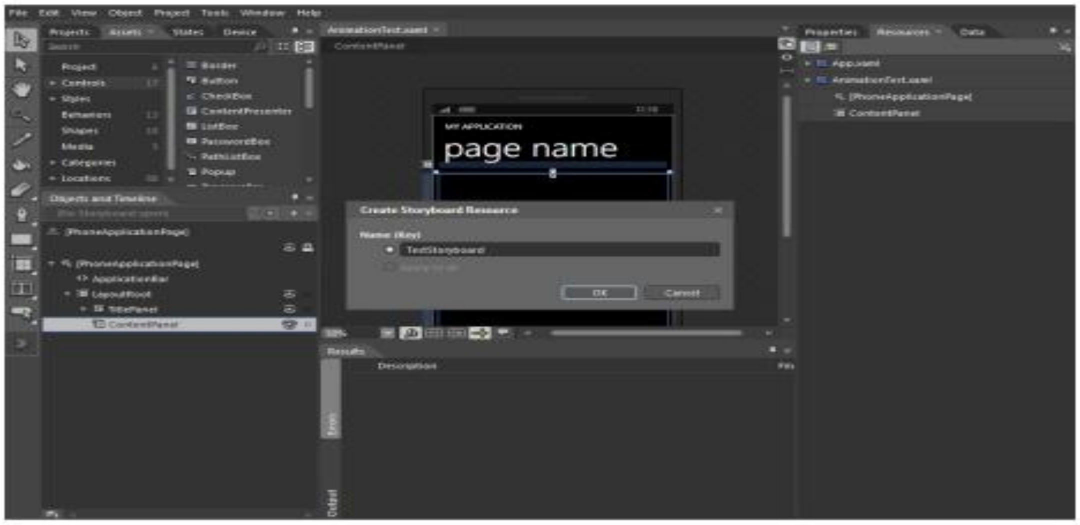

File Edit View Object Project Tools Window Help
Create Storyboard Resource
Name (Key)
TextStoryboard
OK
Cancel
MY APPLICATION
page name

FIGURE 8-9: Storyboards in Blend

d.

2. Pivot vs Panorama: Both the Pivot and Panorama controls are used to delineate categories and subsets of data. With the Pivot control you get strict isolation of these groupings, with the menu providing discoverable UI to show the other categories. 2. With the
Panorama control you get transitions between the groupings with discoverable content on the window boundaries.

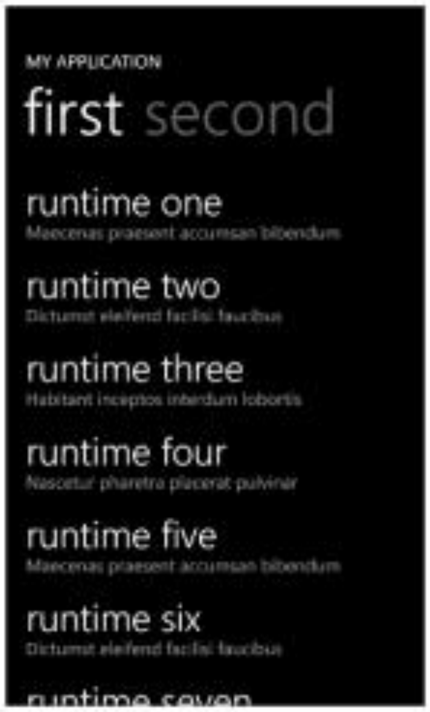

MY APPLICATION
first second
runtime one
Maecenas praesent accumsan bibendum
runtime two
Dictumst eleifend facilisi faucibus
runtime three
Habitant inceptos interdum lobortis
runtime four
Nascetur pharetra placerat pulvinar
runtime five
Maecenas praesent accumsan bibendum
runtime six
Dictumst eleifend facilisi faucibus
runtime seven

FIGURE 8-10: Pivot control

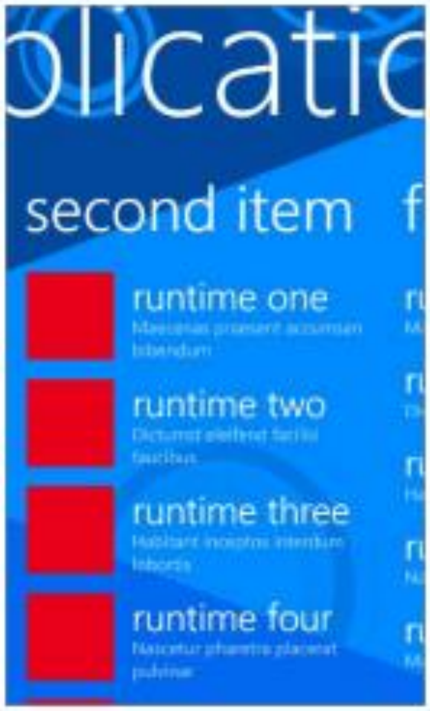

plicatio
second item
runtime one
Maecenas praesent accumsan bibendum
runtime two
Dictumst eleifend facilisi faucibus
runtime three
Habitant inceptos interdum lobortis
runtime four
Nascetur pharetra placerat pulvinar

FIGURE 8-11: Panorama control

The Windows Phone Emulator
The Windows Phone 7 emulator is a very powerful tool. Not just a simulator, the emulator runs a completely sandboxed virtual machine in order to better mimic the actual device. It also comes with some customization and runtime tools to manipulate sensors that are being emulated on the device, including GPS and accelerometer, as well as provide a way to capture screenshots while testing and developing applications. The debugging experience inside of Visual Studio is superior to the ones in Eclipse and the third-party frameworks. The load time of the Emulator is quite fast. It acts responsively, and the step-through just works.
10.b. Write short note on other useful windows phone thing.

IOS applications can be used do many more tasks as explained in following sections.
Offline Storage

Even if your application is using a web service for retrieving information, at some point you may need to save information on the device. Depending on the size and type of data, you have a few different options.
Plist : Property lists are the simplest way to store information on the device. In the Mac world, many applications use the plist format to store application settings, information about the application, and even serialized objects. It’s best to keep the data contained in these files simple and small, though.
The following example finds the path to a plist stored in the supporting files directory, with a name of example. It then loads the plist into a dictionary object, and loops through each time writing the contents of each item in the plist to the debug console.

```
- (void)getValuesFromPlist
{
// build the path to your plist
NSString *path = [[NSBundle mainBundle] pathForResource:
@”example” ofType:@”plist”];
// load the plist into a dictionary
NSDictionary *pListData = [[NSDictionary alloc]
initWithContentsOfFile:path];
// loop through each of the Items in the property list and log
for (NSString *item in pListData)
NSLog(@”Value=%@”, item);
}
```

Core Data: If the data that you need to persist on the device is nontrivial, meaning there is a great deal of it or its complex, Core Data is the way to go. Core Data is described by Apple as a “schemadriven object graph management and persistence framework.” Core Data is not an ORM (Object Relational Mapper). Core Data is an API that abstracts the actual data store of the objects. Core
Data can be configured to store these objects as a SQLite database, a plist, custom data, or a binary file. Core Data has a steep learning curve, but is well worth learning more about if your app will have a great deal of data held within.

GPS
One of the great benefits to mobile devices is the GPS functionality. Once you are able to get over the hurdles of learning the basic functions within the iOS platform, starting to work with the GPS functions can be a great deal of fun. The GPS functions are located in the CoreLocation framework, which is not added to a new project by default. To do this, you will need to click the Build Phases tab on the project settings page.
Once on the Build Phases tab, expand the Link Binary with Libraries section, and click the + button.
You are then prompted with a list of frameworks to add. Select the CoreLocation.framework.
Use the code given below –

```
// GPS Example
locationManager = [[CLLocationManager alloc] init];
locationManager.delegate = self;
locationManager.distanceFilter = kCLDistanceFilterNone;
// get GPS DatalocationManager.desiredAccuracy =
kCLLocationAccuracyHundredMeters;
[locationManager startUpdatingLocation];
```

Notifications
Setting up notifications for Windows Phone 7 is a multistage process. First you must build up a push channel to receive communications within your app. Creating that push channel provides you
with a Service URI to post data to. Posting data in specific formats determines what type of message will be displayed to the client app. There are three types of notifications:

1. Toast notification: The first and simplest is the *toast notification.* With a toast notification
you can pass a title, a string of content, and a parameter.

 The title will be boldfaced then displayed

 the content will follow non-boldfaced, and

• the parameter will not be shown, but it is what is sent to your application when the
user taps on the toast message. This can contain parameters to load on the default
page, or a relative link to the page you want loaded when the app loads as a result of
the tap. Then the user taps on the *toast message.*

2. Tile notification: With the tile notification you can update the application tile content. The
XML data that you post contains fields for the title on

 the front of the tile,

 front of the tile background image,

 the count for the badge,

 the title for the back of the tile,

 the back of the tile background image, and

 string of content for the back of the tile.

3. Raw Notifications: The third and most developer-centric notification type is raw. With the
raw notification type you can pass data directly to the app. It will not be delivered if the
application is not running.

Accelerometer

In addition to GPS, Windows Phone 7 devices are outfitted with an accelerometer. The emulator
provides a 3-D interface for simulating accelerometer change events. You can track the
movement
of the device by capturing the ReadingChanged event on the accelerometer. However, you need
to have a delegate to call back to the UI thread if you want to display anything special based on
the
event. If the application can access the UI thread, the ReadingChanged event handler will call the
delegate function; otherwise, it will dispatch the event on the UI thread. You must also make
sure
that when you are done capturing this data, you stop the accelerometer to preserve battery life.

Web Services

The Derby application is an example of leveraging data over the web to add value to your
application. If you don’t want to be the central repository for all data exposed to your users, you
can
leverage web services that exist from other vendors.