CMR
INSTITUTE OF
TECHNOLOGY

USN

Internal Assessment Test 1 Answer Key– Nov. 2021									
Sub:	Mobile Applications					Sub Code:	18MCA52	Branch:	MCA
Date:	11/11/2021	Duration:	90 min's	Max Marks:	50	Sem	IV		

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

The *Gestalt principles* have had a considerable influence on design, describing how the human mind perceives and organizes visual data. The Gestalt principles refer to theories of visual perception developed by German psychologists in the 1920s. According to these principles, every cognitive stimulus is perceived by users in its simplest form. Key principles include *proximity, closure, continuity, figure and ground,* and *similarity.*

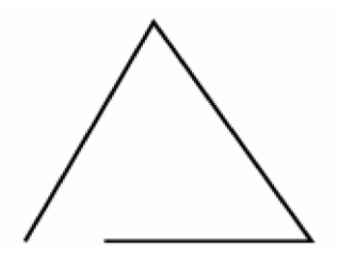
Proximity:

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

- Many smaller parts can form a unified whole.
- Icons that accomplish similar tasks may be categorically organized with proximity.
- Place descriptive text next to graphics so that the user can understand the relationship between these graphical and textual objects.

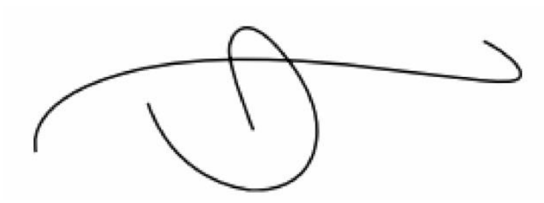
Closure:

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



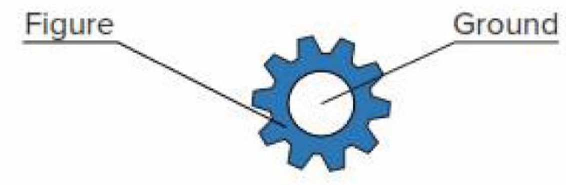
Continuity:

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



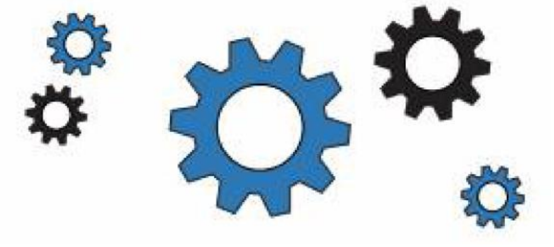
- Smooth visual transitions can lead users through a mobile application, such as a link with an indicator pointing toward the next object and task.

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



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

Similarity:
- Similar elements are grouped in a semi-automated manner, according to the strong visual perception of colour, form, size, and other attributes. Figure 1.5 illustrates it.
- In perceiving similarity, dissimilar objects become emphasized.
- Strict visual grids confuse users by linking unrelated items within the viewport.
- The layout should encourage the proper grouping of objects and ideas.



Q2) What are the preliminary cost involved in mobile application development?
There are many costs associated with mobile application development.
- Each developer will need hardware and software to develop the applications on.
- The team will need devices to test the software on.
- And if you want to deploy your application to any public market, then your company will need accounts on the various markets (these often renew annually).

Hardware

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

Software

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

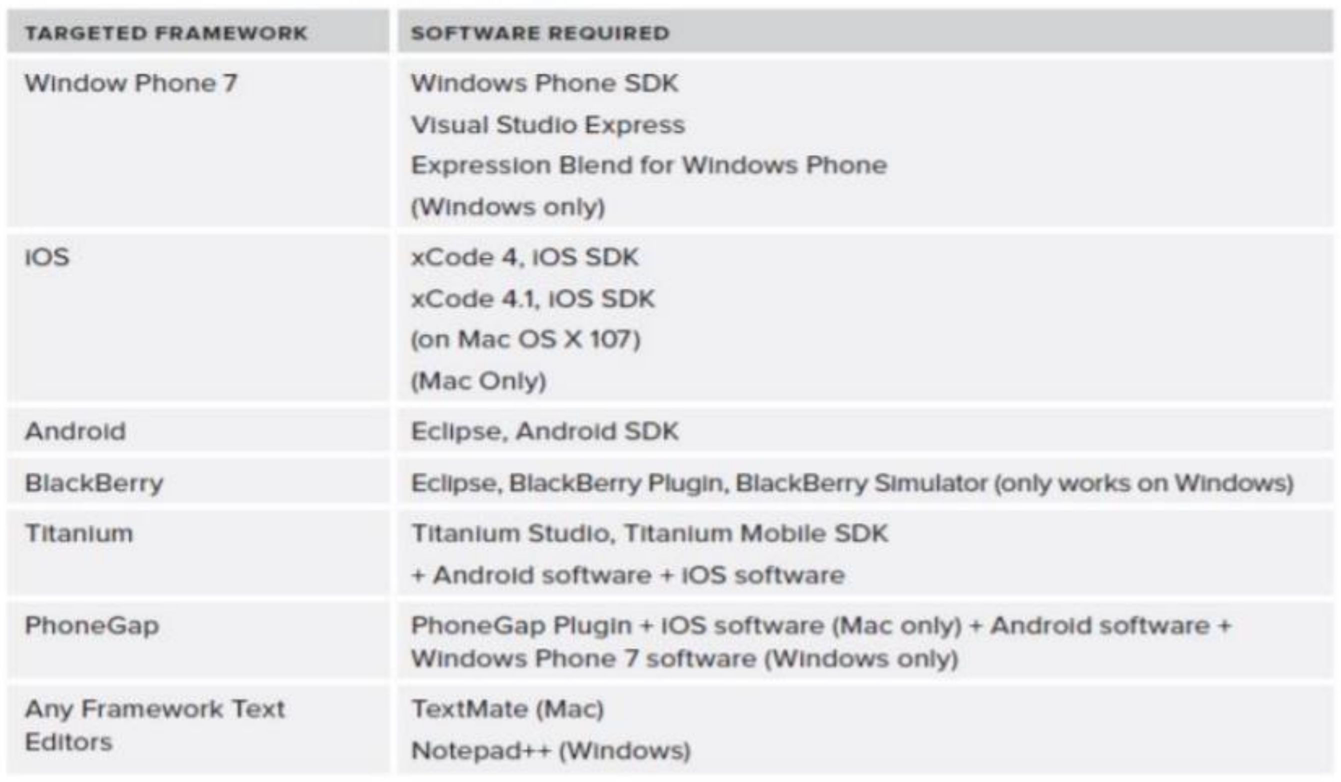
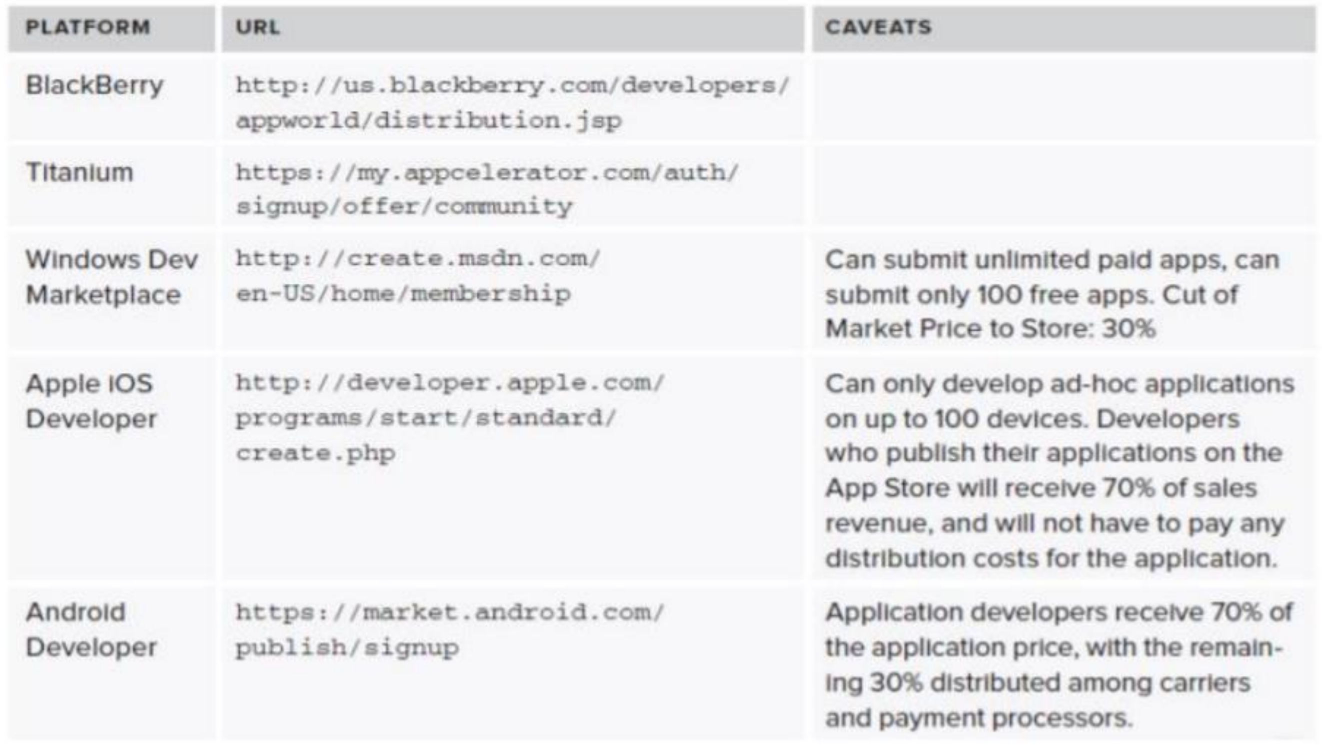


TABLE 1-1: Software Needed for Development

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

Licenses and Developer Accounts

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



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

Documentation and APIs

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

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

The Bottom Line

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

Q3) Describe the effective use of screen real estate

- The first step to use the smaller interfaces of mobile devices effectively is to know the context of use. Who are the users, what do they need and why, and how, when, and where will they access and use information?
- Mobile design is difficult, as developers try to elegantly display a telescoped view of almost limitless information. But user experience issues are amplified on mobile interfaces.
- Cognitive load increases while attention is diverted by the needs to navigate, remember what was seen, and re-find original context.
- Cognitive load is the mental effort to comprehend and use an application, whatever the inherent task complexity or information structure may be.
- Effectively use screen real estate by embracing minimalism, maintaining a clear visual hierarchy, and staying focused.

Embrace Minimalism

- Limit the features available on each screen, and use small, targeted design features.
- Content on the screen can have a secondary use within an application, but the application designer should be able to explain why that feature is taking up screen space.
- Banners, graphics, and bars should all have a purpose.

Use a Visual Hierarchy

- Help users fight cognitive distractions with a clear information hierarchy.
- Draw attention to the most important content with visual emphasis.
- Users will be drawn to larger items, more intense colors, or elements that are called out with bullets or arrows; people tend to scan more quickly through lighter color contrast, less intense shades, smaller items, and text-heavy paragraphs.

- A consistent hierarchy means consistent usability; mobile application creators can create a hierarchy with position, form, size, shape, color, and contrast.

Stay Focused

- Start with a focused strategy, and keep making decisions to stay focused throughout development.
- A smaller file size is a good indicator of how fast an application will load, so the benefits of fighting feature creep extend beyond in-application user experience.
- Focused content means users won't leave because it takes too long for the overwhelming amount of images per screen to load.
- And users won't be frustrated with the number of links that must be cycled through to complete a task. Text-heavy pages reduce engagement as eyes glaze over and users switch to another application.
- If people have taken the time to install and open an application, there is a need these users hope to meet.
- Be methodical about cutting back to user necessities. Build just enough for what users need, and knowing what users need comes from understanding users

Q4) Explain the various information design tools of mobile interface design

Sketching and Wireframes

- Sometimes we need to shape ideas on paper before focusing on the pixels.
- Storyboard application screens to outline features and flow, focusing on the big picture.
- Save wasted time developing the wrong thing the right way by involving all key stakeholders in the sketching and wire framing process.
- Mobile stencils are even on the market to help non doodlers pencil in ideas before turning to computer screens**.**
- A wireframe is a rough outline of each application's framework.
- Stay focused on functionality during wire framing; these easy-to-share, easy-to-edit files are just a skeleton of the design.
- A simple image will do, but tools such as Balsamiq Mock-ups let designers drop boilerplate into a wireframe editor

Mock-up Designs

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

Prototype:

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

On-device Testing:

- One of the most important tools during design will be the physical device.
- Buy, or Borrow, the devices and application will run on.

Simulators and Emulators:

- Simulators and emulators are important when the hardware is unavailable and the service contracts for devices are prohibitively expensive.
- A simulator uses a different codebase to act like the intended hardware environment.
- An emulator uses a virtual machine to simulate the environment using the same codebase as the mobile application.
- It can be cost prohibitive to test on many devices, making emulators incredibly useful.
- Emulators can be run in collaboration with eye-tracking software already available in most testing labs, but an emulator lacks the touch experience of a mobile application.
- At an absolute minimum, use one of the target devices for user testing at this level.
- During design, development, testing, and demonstration, these tools are incredibly valuable.

Q5) Explain Obtaining the IOS tools and SDK.

Important tools to be acquired for IOS App development
are listed in to following categories:

- Hardware
- xCode and IOS SDK
- The IOS Human Interface Guideline

1) Hardware

To develop iPhone, iPod, and iPad applications you must have a Mac. The iPhone SDK runs only on Mac OS X. The only sanctioned hardware for iPhone, iPod, and iPad development is an Intel-based Macintosh.

- Program Levels: If you do not have an Apple Developer account, you can create a free account at https://developer.apple.com/. Having the Apple Developer account allows you to create iOS applications and run them locally on your machine using the iOS Simulator. To deploy applications you have created to a physical device (iPhone, iPad, iPod Touch) you must belong to the iOS Developer program.
  - 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, and you are allowed to name 100 devices within your iOS Developer account.
  - 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. This level of the program will not allow you to distribute apps through the App store, but allows ad hoc distributions (distribute directly to a device without using the App Store) to devices in your organization. A valid Dun & Bradstreet (DUNS) number is required, and this program level will take a little bit longer to get enrolled in. This process takes well over a month before acceptance into the program.
  - 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.

- The IOS Provisioning Portal: iOS Developer Center that allows you to create the files necessary to deploy development and distribution (production) builds onto physical devices.

- Certificates: During the development process of your iOS app, you will more than likely create both a development and a distribution certificate. These certificates are used to digitally sign the app, and verify you are who you say you are.
- App IDs: Each iOS application that you create (that you intend to deploy to a device) needs to be identified on the App IDs section of the iOS Provisioning Portal. The app
- ID that is created is a unique ID that contains a number from Apple and then a bundle identifier that you specify. The bundle identifier is usually in the format com.companyname.appname. As you start to develop more applications, they tend to become messy in this interface.
- Devices: The Devices section in the iOS Provisioning Portal section allows developers to maintain a list of devices in which their iOS applications will be developed. These are the devices that are either used for testing your app or for ad-hoc deployments. The number of devices that you can register on this screen relates to the type of Apple Developer account level you selected. For example, if you registered at the iOS Developer level, you will be able to add 100 devices. This number is 100 per year, meaning if you add 100 devices and then delete 10, you are still out of spaces for accounts until you re-enroll in the program the following year, which will still only have a maximum of 100 devices.
- Provisioning Files: After the certificate, the app ID, and devices have been created/added, you can then create a provisioning profile. The provisioning profile combines the information about which apps/certificates can be installed on which devices. As with certificates there will be a Development and Distribution version.

2) xCode and IOS SDK

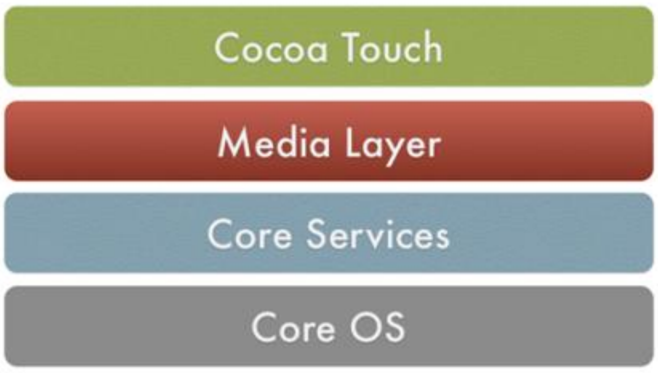
To create native iOS applications, you will need to install both the xCode IDE as well as the iOS SDK. Although you can obtain xCode by using the App Store within Mac OS X. Downloading xCode and the SDK from the downloads section in the iOS Dev Center.

- Installation: After you follow the steps to install xCode, you should have the xCode IDE as well as a great deal of other useful development tools installed to /Developer/Applications. You can start xCode from this directory or by using spotlight.
- Components of iPhone SDK: The iPhone SDK includes a great number of tools that help create iOS for apps. These tools range from debugging and profiling to developing. Following is the list of most common tools that we use that are included in the iOS SDK:
  - xCode: xCode is Apple's Integrated Development Environment (IDE) for creating Objective-C applications. xCode enables you to manage, author, and debug your Objective-C projects.
  - Dashcode: Dashcode is an IDE that enables you to develop web-based iPhone/iPad applications and Dashboard widgets.
  - iPhone Simulator: This tool provides a method to simulate an iPhone or iPad device on your Mac, for use with testing your iOS applications.
  - Interface Builder: The Interface Builder, or IB, is a visual editor that is used for designing the user interface for your iOS application.
  - Instruments: Instruments is a suite of tools that helps you analyze your iOS application and monitor for performance bottlenecks as well as memory leaks in real time while attached to an iOS device or iOS Simulator.

3) The IOS Human Interface Guideline

The iOS Human Interface Guideline (HIG) document is one of the most valuable tools to the iOS developer. The iOS HIG describes guidelines and principles that help the iOS developer design a superlative user interface and user experience. It is very important for new iOS developers to read through this document; if you do not develop using the UI principles found in the HIG, your application could be rejected when submitted to the Apple App Store.

Q6) Explain architecture of IOS



- Lower layers gives the basic services which all application relies on and higher level layer gives sophisticated graphics and interface related services
- Apple provides most of its system interfaces in special packages called frameworks
- A framework is a directory that holds a dynamic shared library that is .a files, related resources like as header files, images, and helper apps required to support that library
- Every layer have a set of Framework which the developer use to construct the applications

Cocoa Touch Layer

- EventKit framework – gives view controllers for showing the standard system interfaces for seeing and altering calendar related events
- GameKit Framework – implements support for Game Center which allows users share their game related information online
- iAd Framework – allows you deliver banner-based advertisements from your app
- MapKit Framework – gives a scrollable map that you can include into your user interface of app.
- PushKitFramework – provides registration support for VoIP apps
- Twitter Framework – supports a UI for generating tweets and support for creating URLs to access the Twitter service
- UIKit Framework – gives vital infrastructure for applying graphical, event-driven apps in iOS. Some of the Important functions of UI Kit framework:
  - Multitasking support
  - Basic app management and infrastructure.
  - User interface management
  - Support for Touch and Motion event.
  - Cut, copy and paste support and many more.

Media Layer

Graphics, Audio and Video technology is enabled using the Media Layer

Layer Graphics Framework . :

- UIKit Graphics – It describes high level support for designing images and also used for animating the content of your views.
- Core Graphics framework – It is the native drawing engine for iOS apps and gives support for custom 2D vector and image based rendering.
- Core Animation – It is an initial technology that optimizes the animation experience of your apps.
- Core Images – gives advanced support for controlling video and motionless images in a nondestructive way
- OpenGl ES and GLKit – manages advanced 2D and 3D rendering by hardware accelerated interfaces

- Metal – It permits very high performance for your sophisticated graphics rendering and computation works. It offers very low overhead access to the A7 GPU.

Audio Framework:

- Media Player Framework – It is a high level framework which gives simple use to a user's iTunes library and support for playing playlists.
- AV Foundation – It is an Objective C interface for handling the recording and playback of audio and video.
- OpenAL – is an industry standard technology for providing audio.

Video Framework

- AV Kit – framework gives a collection of easy to use interfaces for presenting video.
- AV Foundation – gives advanced video playback and recording capability.
- Core Media – framework describes the low level interfaces and data types for operating media.

Core Services Layer

- Address book framework – Gives programmatic access to a contacts database of user.
- Cloud Kit framework – Gives a medium for moving data between your app and iCloud.
- Core data Framework – Technology for managing the data model of a Model View Controller app.
- Core Foundation framework – Interfaces that gives fundamental data management and service features for iOS apps.
- Core Location framework – Gives location and heading information to apps.
- Core Motion Framework – Access all motion based data available on a device. Using this core motion framework Accelerometer based information can be accessed.
- Foundation Framework – Objective C covering too many of the features found in the Core Foundation framework
- Healthkit framework – New framework for handling health-related information of user
- Homekit framework – New framework for talking with and controlling connected devices in a user's home.
- Social framework – Simple interface for accessing the user's social media accounts.
- StoreKit framework – Gives support for the buying of content and services from inside your iOS apps, a feature known as In-App Purchase.

Core OS Layer
The Core OS layer holds the low level features that most other technologies are built upon.

- Core Bluetooth Framework.
- Accelerate Framework.
- External Accessory Framework.
- Security Services framework.
- Local Authentication framework.

Q7) Write a note on other useful windows phone Thing.

1) Offline Storage

- Windows Phone 7 has the System.IO.IsolatedStorage namespace to handle persisting data between application runs. Isolated storage is application-specific storage on the device filesystem.
- The simplest means of implementing an isolated storage solution in Windows Phone is to leverage your PhoneApplicationService's state-based events. Launching and Activated handle application load and resume from tombstone, respectively; Closing and Deactivated handle application exit and tombstone, respectively.
- Making sure that your application loads your isolated storage instance on Launch and Activate, and saves on Close and Deactivate, gives you tremendous capability with little effort.

- The following code persists a unique identifier to be passed to a web service as part of an authentication token. You first need to declare the property for the unique identifier in your App.xaml.cs (your application's code behind file):

```
public partial class App : Application
{
        public Guid UserAuthToken { set; get; }
}
```

2) 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.

3) GPS
Windows Phone 7 has built-in functionality for leveraging the geolocation sensors in your device. Using the System.Device.Location namespace and tracking the PositionChanged event of a GeocoordinateWatcher adds a button to your application bar that will tell you the device's distance from our local derby team, the Lansing Derby Vixens. The Windows Phone emulator has a great interface for mocking GPS location changes while developing and debugging your app.

4) 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.

5) 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.

Q8) 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 behind

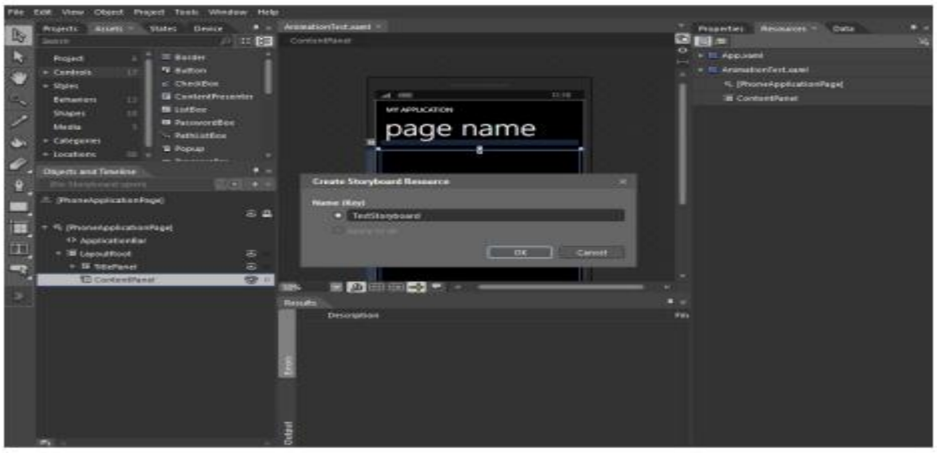


FIGURE 8-9: Storyboards in Blend

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.

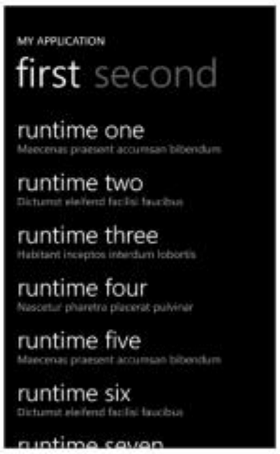


FIGURE 8-10: Pivot control

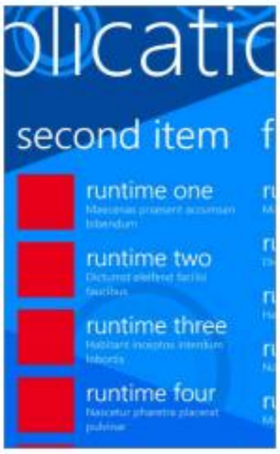


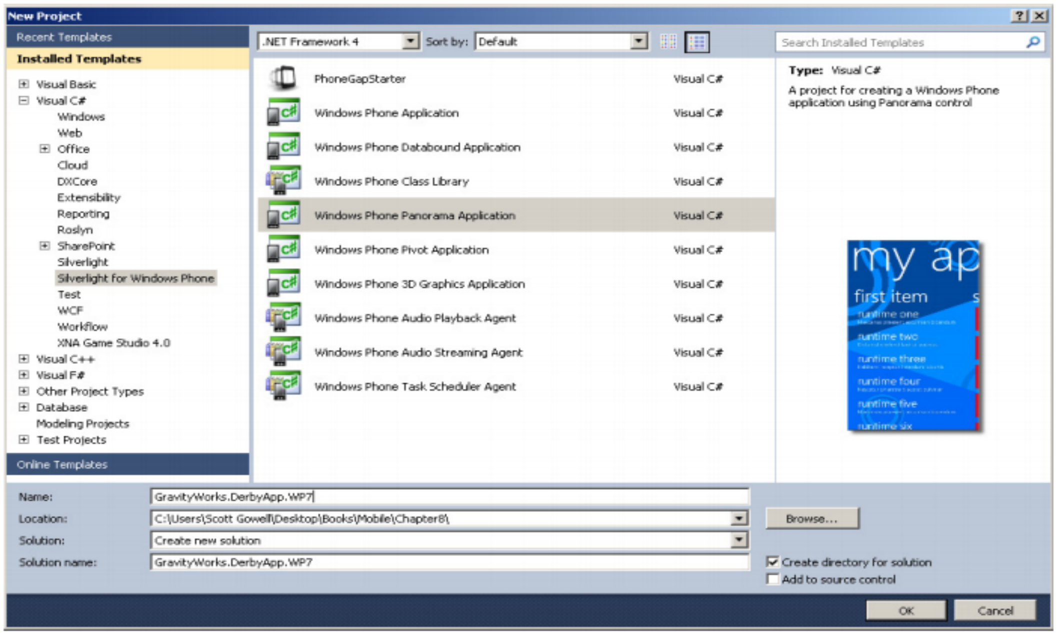
FIGURE 8-11: Panorama control

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.

Q9) How to build derby app in windows phone 7? Explain

Here, you implement the features of the Derby Names project using Microsoft Visual Studio, while also taking time to learn Windows Phone 7–specific technologies.

Creating the Project:
Open Visual Studio and create a new Windows Phone project. For this application, choose Panorama because it offers a UI in which you can share your data.
In a Panorama application the application is created with the default Panorama background.
Visual Studio will create SampleData and ViewModels for your application. Ultimately, you will be able to remove these from your application when you implement your service communications. App.xaml is the entry point for your application and MainPage.xaml is the page that loads by default.


New Project

Recent Templates
Installed Templates
- Visual Basic
- Visual C#
  - Windows
  - Web
  - Office
  - Cloud
  - DXCore
  - Extensibility
  - Reporting
  - Roslyn
  - SharePoint
  - Silverlight
  - Silverlight for Windows Phone
  - Test
  - WCF
  - Workflow
  - XNA Game Studio 4.0
- Visual C++
- Visual F#
- Other Project Types
- Database
- Modeling Projects
- Test Projects

Online Templates

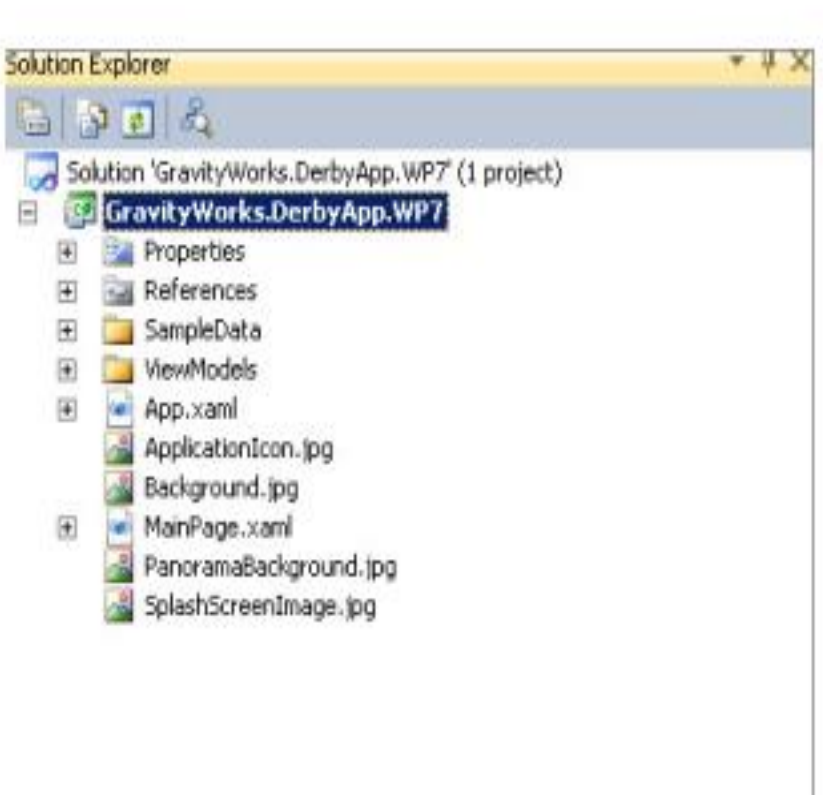
.NET Framework 4 | Sort by: Default

PhoneGapStarter	Visual C#
Windows Phone Application	Visual C#
Windows Phone Databound Application	Visual C#
Windows Phone Class Library	Visual C#
Windows Phone Panorama Application	Visual C#
Windows Phone Pivot Application	Visual C#
Windows Phone 3D Graphics Application	Visual C#
Windows Phone Audio Playback Agent	Visual C#
Windows Phone Audio Streaming Agent	Visual C#
Windows Phone Task Scheduler Agent	Visual C#

Type: Visual C#
A project for creating a Windows Phone application using Panorama control

Name:	GravityWorks.DerbyApp.WP7
Location:	C:\Users\Scott Gowell\Desktop\Books\Mobile\Chapter8\
Solution:	Create new solution
Solution name:	GravityWorks.DerbyApp.WP7

[✓] Create directory for solution
[ ] Add to source control


Solution 'GravityWorks.DerbyApp.WP7' (1 project)
GravityWorks.DerbyApp.WP7
Properties
References
SampleData
ViewModels
App.xaml
ApplicationIcon.jpg
Background.jpg
MainPage.xaml
PanoramaBackground.jpg
SplashScreenImage.jpg

User Interface:
The default Panorama application defines its DataContext in XAML. The DataContext has first item's binding associated by default. The Panorama control can be likened to any collection-based UI element (UITableView in iOS or the ListView in Android), and the PanoramaItems are the respective rows in that collection element. When you feel familiar enough to start working with the data you will need to create a service reference to the Odata feed.

To reference an OData feed you need only to right-click your project, choose Add Service Reference, enter in the URL of your service, and click Go. After it has found the service it should enumerate the models. You are then allowed to update the namespace and create this reference. Once you create the service you can start working with the Panorama control to bind the data available from these entities. After you have made this service, be sure to reference this entity context when your page needs to make calls to the service:
readonly DerbyNamesEntities context = new DerbyNamesEntities(new Uri("http://localhost:1132/DerbyNames.svc/"));

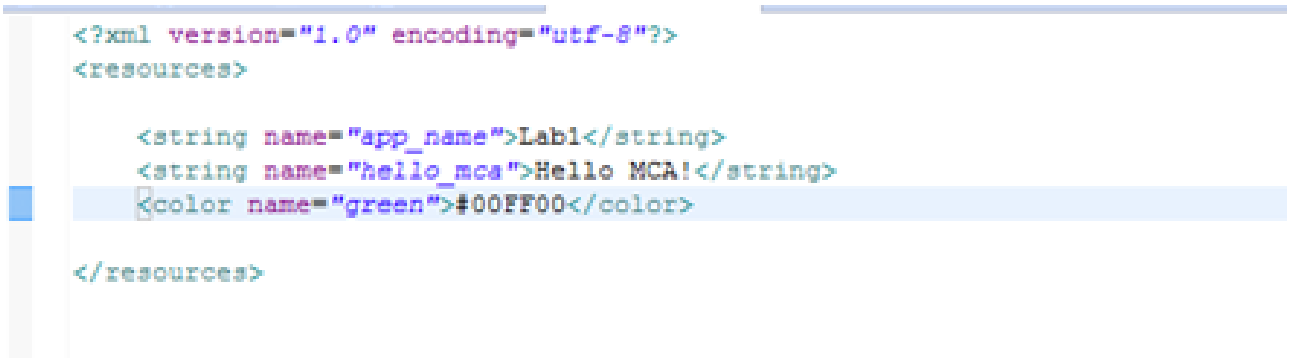
Derby Names:
To bind data to your Panorama item you need to set the ItemsSource and TextBlock bindings. Each individual entry in the DerbyNames entity in OData contains properties for Name and League, which you will bind to the TextBlocks in your Panorama item.

Leagues: Each derby team belongs to a league. The entity for League is similar to the DerbyNames entity, and will make it easy to bind from.

Q10) Develop a mobile application to display a background image of your college and print "Hello MCA" in green color

- Creating a New project: Open Eclipse and then click on File -> New ->Android Application Project.

- Then type the Application name as “Lab1″. Select minimum required SDK , Target SDK and compile with SDK API level as 16 Click Next.
- Then select the Empty Activity and click Next.
- Finally click Finish.
- Import the college image into your project. You can either drag and drop the image or you can copy paste image in any of the drawable folder.
- Open the layout file ‘activity_main.xml’. Go to graphical layout -> select the layout -> click on background property -> select the image from the reference chooser window and click ok.
- Go to strings.xml file and create new string tag with text “Hello MCA” and color tag with green color

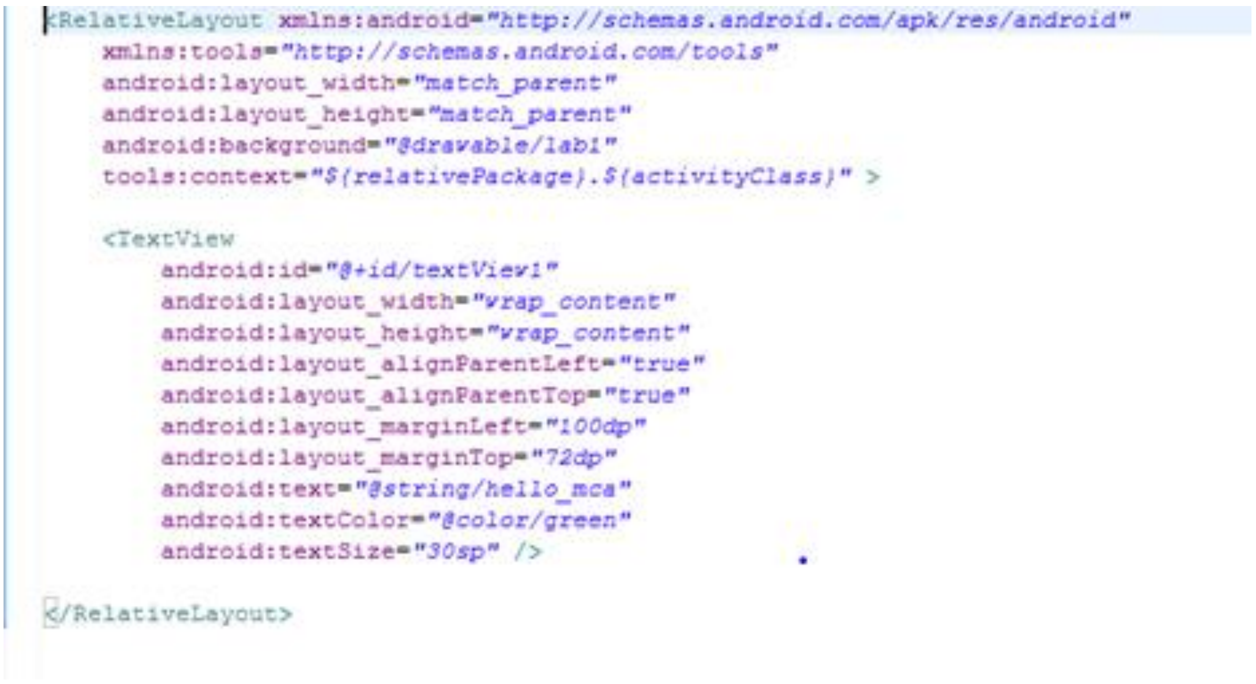


```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Lab1</string>
    <string name="hello_mca">Hello MCA!</string>
    <color name="green">#00FF00</color>

</resources>
```

- Go to activity_main.xml file and change the text, textColor and textSize properties as shown below



```
<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:background="@drawable/lab1"
    tools:context="${relativePackage}.${activityClass}" >

    <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_marginLeft="100dp"
        android:layout_marginTop="72dp"
        android:text="@string/hello_mca"
        android:textColor="@color/green"
        android:textSize="30sp" />

</RelativeLayout>
```

- Run the application to see the output