CMR
INSTITUTE OF
TECHNOLOGY

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

Internal Assessment Test 1 – Sep. 2018

Sub:	Mobile Application					Sub Code:	16MCA53	Branch:	MCA
Date:	11/09/2018	Duration:	90 min's	Max Marks:	50	Sem	V		OBE

1 List and Explain all the costs associated with mobile application development [10]

There are many costs associated with mobile application development.

- Each developer will need hardware and software to develop the applications on.
- The team will need devices to test the software on.
- And if you want to deploy your application to any public market, then your company will need accounts on the various markets (these often renew annually).

Hardware

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

Software

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

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.

2 Explain in detail how you can effectively use screen real estate. [10]

- 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

3 List and explain all the Gestalt key principles. [10]

Gestalt principles include proximity, closure, continuity, figure and ground, and similarity.

Proximity

FIGURE 4-1: Proximity

- Users tend to group objects together. Elements placed near each other are perceived in groups; as shown in Figure, people will see one group of three gears, and one group of two gears.
- 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

FIGURE 4-2: 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.
- Figure illustrates the concept of closure: people recognize a triangle even though the figure is not complete.
- Harness the closure concept to create icons with a strong primary silhouette, without overloading users on pixilated and overdone details.
- In grid patterns with horizontal and vertical visual lines, use closure to precisely show the inside and outside of list items.

Continuity

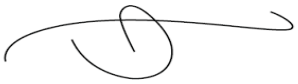


FIGURE 4-3: Continuity

- 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.
- When people see Figure 4-3, they perceive the horizontal stroke as distinct from the curled stroke, 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

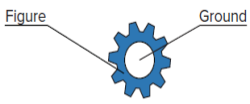


FIGURE 4-4: Figure and ground

- A figure, such as a letter on a page, is surrounded by white space, or the ground. In Figure 4-4, the figure is the gear icon, and the ground is the surrounding space.
- Complex designs can play with the line between "figure" and "ground," but mobile interfaces speed user frustration with unclear distinctions.
- Primary controls and main application content should maintain a distinct separation between figure and ground.

Similarity

FIGURE 4-5: Similarity

- Similar elements are grouped in a semi automated manner, according to the strong visual perception of color, form, size, and other attributes (see Figure).
- 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.

4 Explain following usability considerations [10]

i. Determining and Reaching the Target Audience

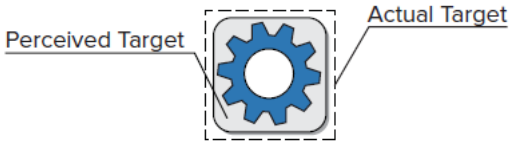
ii. Designing for Gestures

iii. Error Protection and Correction.

Determining and Reaching the Target Audience

- Research and determine the target audience: Who are they, what do they need, and how can they get it?
- It is important to consider the different hardware and usage patterns, whether holding an iPad with two hands in a meeting or thumbing through menus on the bottom of an Android phone screen.
- Mobile applications can connect people with the world around them, providing personalized functionality. Popular applications include social networking; weather or traffic; consumable content such as music or news; productivity, education, and enrichment; and games.
- Usable mobile applications help users perform tasks that are appropriate for mobile devices.
- Mobile tasks can involve quickly finding current information under some deadline (perhaps even an emergency), often requiring privacy and communication with other people.
- Usability therefore starts during mobile strategy, when stakeholders determine that the target audience will use the application functionality on mobile devices.

Designing for Gestures



- If it is difficult to discover gestures to tap, pinch, swipe, and zoom across an application, this means average users will be missing out on those great features developers spent time building.
- One simple solution is a pop-up box to announce the first time each gesture-prompted feature becomes available.
- Avoid swipe ambiguity: user error rates will be higher if the same swipe gesture can prompt multiple actions on the same screen.
- Be sure to consider the perceived versus the actual target area when designing for mobile gestures.
- The actual target area for touch input may be larger than the perceived target area as seen on the screen as shown in Figure. When there is no nearby gestural action, accommodate for finger sizes and user error by extending the actual input area into the white spaces beyond buttons and tabs.
- As much as possible, design dead space to reduce errors from touching nearby target areas.

Error Protection and Correction

- Without a mouse, touch-screen navigation through menus, buttons, links, and scrolling is more prone to user errors. Be sure to include an “undo” or “back” option.
- Every process should be designed to protect user input.
- Develop a log to preserve input history, and design the interface to simplify information retrieval. It can be difficult, and it is usually frustrating, to reproduce time-consuming data lost to accidental destruction, whether data is lost by user error or otherwise.
- Implicit data protection avoids information loss, such as a hard stop between taps to slow text deletion.
- Explicit protections recover data with an undo option, and abandoned forms can be repopulated with recently entered data.
- Save data as often, and in as much detail, as possible. Because mobile users become easily distracted or bored, always be prepared to stop application processes.

5 Explain in detail all application features of Navigation in Design Patterns [10]

Design Patterns

A design pattern recycles and repurposes components, reusing the best ideas. More than time efficiency, patterns have been refi ned by use. Look to patterns, and maintain a pattern library that works for you, but look to the user and the purpose of an individual design above any best practices

Navigation

Good design makes it clear how users can move through and use application features.

1. Annunciator Panel

   An annunciator panel, seen at the top of Figure, gives information on the state of a mobile device. Though each mobile device will provide a slightly different panel, developers can modify or suppress the annunciator panel — which lists the hardware features such as network connection and battery power — within an application. Because the information in this area is only notifi cations, application users will not usually have any direct interaction with the annunciator panel.

2. Fixed Menu

   A menu that remains fixed to the viewport as users roam content is useful in many situations:

   ➢ When users need immediate access to frequently selected functionality on multiple screens

   ➢ When a revealable menu is already used on the same screen

   ➢ When a lack of controls, confl ict with key interactions, or low discovery makes a revealable menu a poor choice

   Because fixed menus are always accessible, users can interact with page content as needed; keep in mind the small screen real estate, and limit any fixed menus to the absolute necessities.

   Do not stack multiple fixed menus on top of each other, and reconsider the size and scope of an application if you have a fixed menu at both the top and bottom of a screen. The style,

such as whether the menu goes across the top or bottom of the viewport, will be largely determined by the mobile device.

3. Expandable Menu
   When all function options for one page cannot fit on the viewport, an expanding menu can provide selectively available options: a full menu similar to the one shown in Figure will reveal once prompted. A gesture, like a swipe or double tap, may prompt the reveal as well as selecting an on-screen icon. Soft keys — hardware buttons connected to on-screen labels — may also prompt a menu to reveal. Users often benefit from multiple access options, especially for the harder-to-find gestural prompts.
   Users may exit a menu using a back button, a close button that is part of the revealed menu list, or by toggling with the same gesture or soft key that prompted the reveal. Try to keep close functionality on the screen while the menu is revealed.

4. Scroll
   - It is best to limit scrolling, limiting application screens to the size of the viewport whenever possible. Some types of apps you develop will require scrolling to present data to the user effectively, such as an email client. When scrolling must occur, check that the design communicates the area that can be scrolled, the current context of a user, and the ratio of the current view to the total content available
   - Only in-focus items should be able to scroll. Make an application more immersive, incorporating gestures such as tilting to scroll through content. Make sure that scrolling takes place only on a single axis, if possible. When scrolling must occur both horizontally and vertically, consider providing a thumbnail of the user's place within the entire scrolling area.
   - The vertical list simply displays textual information, and is the foundation of information display on many mobile devices.
   - Graphical data — profile photos, category icons, status indicators — can clarify content lists. Use position as well as size, contrast, and other identifiers to show the visual importance of elements that users utilize to sort information.
   - Thumbnails can be replaced by a default icon when a specific image is not available, but the value of the graphical indicator diminishes as more icons are not individually identifiable. Use icons and images to emphasize clarity and categorical distinction — embracing a strong and varied silhouette — over personality or generic graphics.
   - An expandable list, reveals additional, valuable content for selected (touched) list items without leaving the current view.
   - New information can be revealed as an animation whenever possible, aligning users with the structure and context of the expanded content areas. Higher-priority information, whether the revealed content or the list item title, should be set apart with size, color, or contrast.
   - Users may scroll by gesture, device tilt, or on-screen buttons. Choose a different pattern if live-scrolling — a pixel-by-pixel response to user input — is not possible.

5. Notifications and Feedback
   If the user needs new information, application creators must use notifications. These prompts pause ongoing tasks, and enable users to change processes and behaviors according to the new information or feedback.
   Notification can inform a user (presenting a single button to close the notification), offer the ability to cancel a process or correct an error, or give multiple options from which to select. A user should never be presented with more than three options from any notification

6 Explain different types of notifications in Windows phone 7 [10]

- Setting up notifi cations 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 specifi c formats determines what type of message will be displayed to the client app.

You can use 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 when displayed, the content will follow nonboldfaced, and the parameter will not be shown, butit 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.

2. **Tile notification**
   The second and more complex notification is the 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 notification**
   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.

7 List and explain all the tools needed for ios mobile application development. [10]

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.
If you are having a hard time justifying the cost of the hardware, we have had great luck with getting refurbished machines direct from Apple at the following URL:
http://store.apple.com/us/browse/home/specialdeals/mac

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. This is where the money comes in. These programs and prices change from time to time

.

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.

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

No matter which level of Apple Developer program you registered for, you will have access to the iOS Provisioning Portal. This is the section of the 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. The iOS Provisioning Portal Certificate section, found within the iOS developer account web interface; here both development and distribution certificates are created.

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

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.

xCode and the iOS SDK

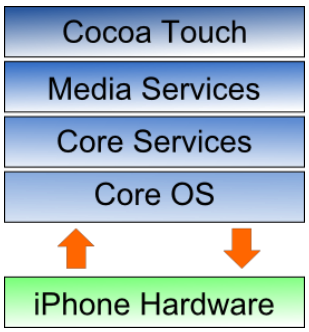
To create native iOS applications, you will need to install both the xCode IDE as well as the iOS SDK. Although you can obtain xCode by using the App Store within Mac OS X, we recommend downloading xCode and the SDK from the downloads section in the iOS Dev Center

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

8 Discuss architecture of IOS with the neat diagram [10]

When we develop apps for the iPhone Apple does not allow us direct access to any of this hardware. In fact, all hardware interaction takes place exclusively through a number of different layers of software that act as intermediaries between the application code and device hardware. These layers make up what is known as an operating system. In the case of the iPhone, this operating system is known as iOS.

The iOS 4 Architecture iOS consists of a number of different software layers, each of which provides programming frameworks for the development of applications that run on top of the underlying hardware.



Cocoa Touch Layer

It is a top layer of the iPhone OS stack and it contains the frameworks that are most commonly used by iPhone application developers. Cocoa Touch is primarily written in Objective-C, and it is based on the standard Mac OS X Cocoa API.

- 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:
    - Basic app management and infrastructure.
    - Multitasking Support

- User interface management
- Support for Touch and Motion event.
- Cut, copy and paste support and many more.

Media Layer

It is the second layer from the top of the stack. It provides the iPhone OS with audio, video, animation and graphics capabilities. As with the other layers of the iPhone OS stack, the Media layer comprises a number of frameworks that can be utilized when developing iPhone apps.

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 controling 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

It is the third layer from the top of the stack. The iPhone Core Services layer provides much of the foundation on which the above layers are built.

Some of the Important Frameworks available in the core services layers are detailed:

- 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 is the bottom layer of the iPhone OS stack and sits directly on top of the device hardware. This layer provides a variety of services including low level networking, access to external accessories and the usual fundamental operating system services such as memory management, file system handling and threads.

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

iPhone Hardware

Hardware devices are managed by iPhone OS and provides the technologies needed for implementing native applications on the phone. The OS ships with several system applications such as Mail, Safari, Phone, that provide standard services to the user.

9 a) Discuss the process of distributing applications in the App Hub (IOS). [5]

To distribute applications in the App Hub you must create a developer account at https://users.create.msdn.com/Register.
Registration costs $99 per year and allows you to:

- Make free, paid, or ad-funded apps and games.
- Submit unlimited paid apps to Windows Phone Marketplace.
- Submit up to 100 free apps to Windows Phone Marketplace; additional submissions are $19.99 USD per submission.
- Expand your reach with worldwide distribution and trial options.

Additionally, all apps are content and code-certified.
The Student account type has specific requirements.
Microsoft DreamSpark (http://www.dreamspark.com/Product/Product.aspx?productid=26) is a program for students, and gives them a free App Hub account.
Submitting your application to the App Hub is a five-step process.

1. First, you upload your compiled application XAP file. The XAP file is the binary for your application that will be pushed to the phone. To do this you must have a unique name for your application, you must select whether this application is being released to the public or simply being distributed to the App Hub for a private beta test, and you need to specify a version for your app.
2. Next, you must provide an application description (this includes category of app, keywords, detailed description, languages supported, and art assets).
3. Third, you set up your price and select what markets you want to distribute your application in.
4. Next, you provide test information so that the developer in charge of approving your app understands the use cases.
5. Finally, you choose your publishing options (as soon as approved, as soon as approved but hidden, manual publish). You then submit your application for certification.

9 b) Discuss all the Components of the iPhone SDK. [5]

The iPhone SDK includes a great number of tools that help create iOS for apps. These tools range from debugging and profiling to developing. This section lists the 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. This product is not in the scope of this book as it is considered an advanced topic.

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.

10 Explain following other useful things in windows phone 7 [10]

i. Offline Storage

ii. GPS

i) 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 file system.
- The simplest means of implementing an isolated storage solution in Windows Phone is to leverage your Phone Application Service's state-based events. Launching and Activated handle application load and resume from tombstone, respectively;

- Closing and Deactivated handle application exit and tomb stoning, 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.
- **Windows Phone 7 Isolated Storage Explorer**
  Available on CodePlex, the Isolated Storage Explorer includes a WPF desktop application and a Visual Studio plug-in to allow developers to manage data held in isolated storage on the device.
  By adding a reference to the Isolated Storage Explorer Assembly and adding a command in your application launching event you get a per-app instance treating your isolated storage like a folder in Windows.

ii) 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

```
GeoCoordinate DerbyVixensLocation = new GeoCoordinate(42.7337, -84.5469);
GeoCoordinateWatcher _GeoCoordinateWatcher;
private void DistanceToVixens()
        {

                try
                {
                        _GeoCoordinateWatcher =
                        new GeoCoordinateWatcher(GeoPositionAccuracy.High)
                        {
                                MovementThreshold = 10 /* 10 meters. */
                        };
                        _GeoCoordinateWatcher.PositionChanged +=
                        GeoCoordinateWatcherPositionChanged;
                        _GeoCoordinateWatcher.Start();
                }
                catch
                {
                }
        }
 private void GeoCoordinateWatcherPositionChanged(object sender,
GeoPositionChangedEventArgs<GeoCoordinate> e)
{
        _GeoCoordinateWatcher.PositionChanged -=
        GeoCoordinateWatcherPositionChanged;
         GeoCoordinate current =
        new GeoCoordinate(e.Position.Location.Latitude, e.Position.Location.Longitude);
        var metersFromVixens = current.GetDistanceTo(DerbyVixensLocation);
         MessageBox.Show(string.Format("{0:0.00} meters from the Lansing Derby Vixens",
         metersFromVixens));
        _GeoCoordinateWatcher.Stop();
        _GeoCoordinateWatcher.Dispose();
        _Geo
}
```