PES INSTITUTE OF TECHNOLOGY (BSC)
V MCA, First IA Test, August 2019
Mobile Applications (17MCA53)
Solution Set

Faculty: Jeny Jijo

1) Explain the cost considerations associated with mobile application development **[8]**

There are many costs associated with mobile application development. Each developer will need

hardware and software to develop the applications on.

a)Hardware

BlackBerry (6 or 7): BlackBerry Bold 9900

- 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

b)Software(Explain the environments for each framework)

TARGETED FRAMEWORK

Window Phone 7
iOS
Android
BlackBerry
Titanium
PhoneGap
Windows
Any Framework Text
Editors

c) Licenses and Developer Accounts

BlackBerry

Titanium

Windows Dev Marketplace - Can submit unlimited paid apps, can submit only 100 free apps. Cut of Market Price to Store: 30%

Apple iOS Developer - 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 - Application developers receive 70% of the application price, with the remaining 30% distributed among carriers and payment processors

d) Documentation and APIs

MSDN Library
iOS Documentation
BlackBerry Documentation
Android SDK Documentation
PhoneGap Documentation

2 Explain the Benefits of mobile app [8]

Make Use of the Native Devices Features

It will always be easier to stretch the hardware boundaries of a mobile app. Great features such as in-app purchasing do not have the same tight integration with the UI and operating system unless you are creating a native app

Offline Content

Many business apps need to display changing data to the user. Depending on the business domain, a mobile web app may not be a good idea. When having the data stored locally, you should also define a strategy on how that offline data is going to be updated. Is it updated when the app is updated through the market or perhaps there a background service that checks to see if the device has access to the Internet, and prompts the user if they would like to obtain a refreshed set of data

- **Richer User Experience**

Users look for apps that have a UI that is consistent with the rest of the apps on their device. It is possible to create HTML and CSS that provide these interfaces on a mobile web app, but it can get difficult.

Many developers opt for creating interfaces that do not resemble iOS, Android, Windows Phone 7, or BlackBerry. It's a design the developer creates on their own. Such a design strategy can work, as long as the correct amount of user interface research has been performed. In most cases, however, it's best to just stick with the UI you should be working with, which is the native UI for the platform

- **Ease of Discovery**

Markets provide a place to present your app to the world. Most users are not using a search engine to fi nd apps for their mobile devices; they are using the built-in search tools within the installed market tool.

Push Notifications

An instant notification on your mobile device means an immediate response is expected. Push notifications simulate the same behavior of text messages, but are app based. Push notifi cations alert users of something that they should be aware of instantly: a new e-mail, a new tweet, or some other bit of information that may be important to the app that was downloaded.

Increased Customer Feedback

Businesses often hope to build brand loyalty through apps. When loyalty has been achieved, you can capitalize on this loyalty within the app, asking for feedback about your company. Quick polls, short forms, and rich integration with social media services such as Facebook and Twitter can provide a level of feedback that is not seen with mobile web apps.

3(b) Explain the difference between mobile web and mobile app [4]

A mobile website is similar to any other website in that it consists of browser-based HTML pages that are linked together and accessed over the Internet (for mobile typically WiFi or 3G or 4G networks). The obvious characteristic that distinguishes a mobile website from a standard website is the fact that it is designed for the smaller handheld display and touch-screen interface. Increasingly, responsive web design is becoming the new standard for websites that are not only mobile-friendly, but that can scale to any sized device - from desktop down to tablet and handheld smartphones.

Like any website, mobile websites/responsive sites can display text content, data, images and video. They can also access mobile-specific features such as click-to-call (to dial a phone number) or location-based mapping.

Apps are actual applications that are downloaded and installed on your mobile device, rather than being rendered within a browser. Users visit device-specific portals such as Apple's App Store, Android Market, or Blackberry App World in order to find and download apps for a given operating system. The app may pull content and data from the Internet, in similar fashion to a website, or it may download the content so that it can be accessed without an Internet connection.

4. Explain about Gestalt Principals with suitable diagrams? [8]

Key principles of Gestalt

Proximity

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

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. 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. 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

- Similar elements are grouped in a semiautomated manner, according to the strong visual perception of color, form, size, and other attributes .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.

The Social Aspect of Mobile

- Social networking and social media outlets collect and distribute chunks of content from people across the globe, adding value to the user experience while spreading ideas and building reputations through trusted social networks.

Connect with Existing Outlets

- **Usability**
  - In a perfect world, usability considerations would be a regular, ongoing part of the whole process, checking how pixels and fingers interact in the real world (many a button has looked great in the mock-up design, only to be too small for actual people to use). This star life cycle is optimal: with evaluation as the center of the star, and various design and development tasks as branches of that evaluation process, this encourages ongoing iterations as user needs are discovered.
- **Accessibility**
  - An application that is easier for people to use with poor or diminished vision, limited dexterity, or a cognitive impairment will be easier for all people to use.
  - **Mobile Web Best Practices (MWBP)** is a World Wide Web Consortium standard defining a set of five checkpoints for mobile accessibility:
    - **Overall Behaviour**
    - **Navigation and Links:**
    - **Page Layout and Content**
    - **Page Definition**
    - **User Input**
- The POUR principles were created for mobile web interfaces, but apply to all mobile viewports and mobile user experiences.
  - **Hearing**

    For moderate hearing loss, adjustable volume control offers a simple way to connect with mobile content. However, most solutions are focused on visual alerts: incoming or missed messages in call logs, text messages, on-screen prompts, and hardware-specific features such as blinking alert lights.
  - **Vision**

    People with low vision through complete blindness may benefit from sliding or flipping a phone to answer and end calls (rather than a touch-screen button), and are likely to consider the hardware first. Popular and industry-standard devices without a flip or slide may be modified to meet the needs of low-vision users. Mobile application creators can consider adjustable font sizes, color contrast, and backlit displays.

5 ***Briefly explain how effectively the screen real estate is used in mobile application development.*** ***[8]***

A design pattern recycles and repurposes components, reusing the best ideas. More than time efficiency, patterns have been refined by use.

- **Navigation**
  - Good design makes it clear how users can move through and use application features
- ***Annunciator Panel***

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

- ***Fixed Menu***

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

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

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

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

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

- Expandable Menu

  – When all function options for one page cannot fit on the viewport, an expanding menu can provide selectively available options. 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.

- ***Scroll***

  – It is best to limit scrolling, limiting application screens to the size of the viewport whenever possible. 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.

- ***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.

  – Feedback is the user-perceived result of an interaction, providing immediate confirmation like a color change, message, or being led to a new page. Delayed feedback leads to user frustration and redundant, often error-inducing input; confirmation feedback is useful when user data could otherwise be lost, and should be indicated with a distinct change in state from the initial view.

6) Explain the mobile application platforms a) Android b) iOS c) BlackBerry OS [8]

1) **Android**

- Android has a diverse ecosystem, with fewer institutionalized restrictions and a wider variety of mobile devices than other popular systems.
- Development of the Android operating system is led by Google, and backed by a global and growing user base

Interface Tips

- Android convention is to place view-control tabs across the top, and not the bottom, of the screen.

Use the main application icon for temporal, hierarchical navigation, instead of a "back" button and main icon link to the home screen

- Don't mimic user interface elements or recycle icons from other platforms.
- Parallax scrolling is common in Android applications.
- Android development can extend to home-screen "widget" tools.

Accessibility

- Google provides guidelines and recommendations, such as testing with the often-preinstalled and always-free TalkBack. Accessibility design guidelines are listed on the Android Developer website

2) **iOS**

- Apple maintains strict design standards, which are detailed and updated online. iOS interface documentation and general mobile design strategies are available from Apple, including design strategies

Interface Tips

- iPhone users generally hold from the bottom of the device, so main navigation items should be in reach of user thumbs.
- Target areas for controls should be a minimum of 44 x 44 points.
- Support standard iOS gestures, such as swiping down from the top to reveal the Notification Center.
- Larger iPad screens are great for custom multi-finger gestures, but non-standard gestures should never be the only way to reach and use important features.

Accessibility

- Accessible touch and gestural controls are available on the iPad and latergeneration iPhones; screen magnifi cation and color contrast adjustments are also available.

3) **BlackBerry OS**

- BlackBerry OS is often the mobile device of choice in government or corporate environments.
- BlackBerry includes native support of corporate emails; and runs on many devices with hard keypads, which is favored by users with accessibility issues as well as late adopters to touch-screen interfaces.

Interface Tips

- Use BlackBerry UI components, not the tabs or other components of alternate platforms.
- Use standard interaction behaviors for an intuitive experience.
- Link common tasks to the BlackBerry track pad according to standard actions:

Press the track pad: Default option, like revealing the menu

  - **Press and hold track pad:** Activate available pop-up box
  - **Press track pad and select Shift:** Highlight content
  - **Press track pad and select Alt:** Zoom
  - **Move finger along track pad:** Cursor or mouse will move accordingly

Accessibility

- BlackBerry mobile devices include text-based push-delivery messages, closed captions on multimedia content, and hearing-aid compatibility for hearing accessibility issues.
- Low-vision users can use the Clarity theme and other screen adjustments, and benefit from tactile keyboards.
- Predictive text and AutoText aid users with mobility and cognitive issues.

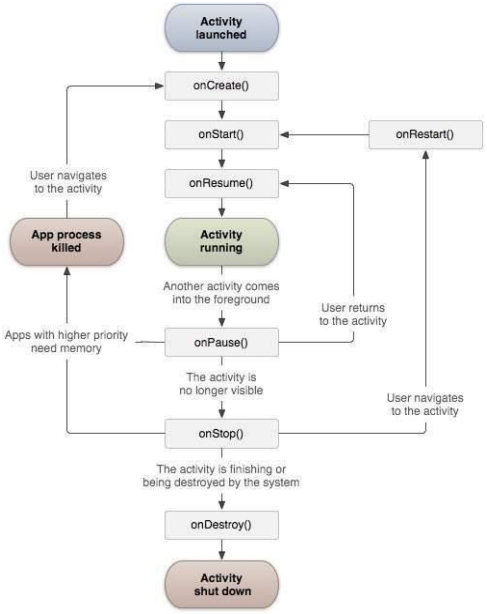
Windows Phone 7

- Developed by Microsoft, Windows Phone 7 (WP7) is a currently smaller contender, focused on consumer markets. Using the “Metro” theme, features are divided into “Live Tiles” that link to applications.

Interface Tips

- WP7 uses movement over gradients for on-screen elements to immerse users in the application experience.
- Users will enter a WP7 application from a “tile,” which can display dynamic and real-time information. Tile images should be in the PNG format, 173 pixels 173 pixels at 256 dpi.
- Do not use a “back” button to navigate back the page stack. All WP7 devices have a dedicated hardware button that should always be used instead.

7) Explain the lifecycle of an activity in Android Applications ***[8]***



- **onCreate()**
  - This is the first callback and called when the activity is first created.
- **onStart()**
  - This callback is called when the activity becomes visible to the user.
- **onResume()**
  - This is called when the user starts interacting with the application.
- **onPause()**
  - The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.
- **onStop()**
  - This callback is called when the activity is no longer visible.
- **onDestroy()**
  - This callback is called before the activity is destroyed by the system.
- **onRestart()**
  - This callback is called when the activity restarts after stopping it.

  An activity class loads all the UI component using the XML file available in *res/layout* folder of the project. Following statement loads UI components from *res/layout/activity_main.xml file*:

  setContentView(R.layout.activity_main);

8) Describe the anatomy of an Android Application ***[8]***

- The **Android Manifest is the heart of your application**. It holds the entire configuration of your app, the permissions you request, the application attributes, and links to instrumentation to be attached to your app. You can edit this in Eclipse's Manifest Editor or in XML because that is how it is saved.

- The AndroidManifest.xml file contains detailed information about the application:
  - It defines the package name of the application as net.learn2develop.HelloWorld.

- The version code of the application is 1. This value is used to identify the version number of your application. It can be used to programmatically determine whether an application needs to be upgraded.
- The version name of the application is 1.0. This string value is mainly used for display to the user.
- The application uses the image named icon.png located in the drawable folder.
- The name of this application is the string named app_name defined in the strings.xml fi le.
- There is one activity in the application represented by the MainActivity.java file.
- 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.

9) What are the components of Android Applications? Explain

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact.

There are following **four main components** that can be used within an Android application

- **Activities**
  - They dictate the UI and handle the user interaction to the smart phone screen. An activity represents a single screen with a user interface, in-short Activity performs actions on the screen.
- **Services**
  - A service is a component that runs in the background to perform long-running operations. Eg- a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.
- **Broadcast Receivers**
  - Broadcast Receivers simply respond to broadcast messages from other applications or from the system.
  - Eg-applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use
- **Content Providers**
  - A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver* class. The data may be stored in the file system, the database or somewhere else entirely

Additional Components

- Fragments
- Views
- Layouts
- Intents
- Resources
- Manifest

10) Explain

a) Fragments

A **Fragment** is a piece of an activity which enable more modular activity design.

- A fragment has its own layout and its own behaviour with its own life cycle callbacks
- We can add or remove fragments in an activity while the activity is running.
- You can combine multiple fragments in a single activity to build a multi-pane UI.
- A fragment can be used in multiple activities.
- Fragment life cycle is closely related to the life cycle of its host activity which means when the activity is paused, all the fragments available in the activity will also be stopped.
- A fragment can implement a behaviour that has no user interface component.
- You create fragments by extending **Fragment** class and You can insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a **<fragment>** element.

Types of Fragments

Basically fragments are divided as three stages as shown below.

- **Single frame fragments** − Single frame fragments are using for hand hold devices like mobiles, here we can show only one fragment as a view.
- **List fragments** − fragments having special list view is called as list fragment
- **Fragments transaction** − Using with fragment transaction. we can move one fragment to another fragment.

b) Action Bar

The toolbar bar (formerly known as action bar) is represented as of Android 5.0 via the Toolbar view group. It can be freely positioned into your layout file. It can display the activity title, icon, actions which can be triggered, additional views and other interactive items. It can also be used for navigation in your application.

To use the toolbar on such devices add a compile dependency to com.android.support:appcompat to your Gradle build file. For example:

compile 'com.android.support:appcompat-v7:22.2.0`

Entries in the toolbar are typically called ***actions***. While it is possible to create entries in the action bar via code, it is typically defined in an XML resource file.

Each menu definition is contained in a separate file in the **res/menu** folder. The Android tooling automatically creates a reference to menu item entries in the **R file**, so that the menu resource can be accessed.

An activity adds entries to the action bar in its **onCreateOptionsMenu()** method.

The **showAsAction** attribute allows you to define how the action is displayed.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
   <item
      android:id="@+id/action_refresh"
      android:orderInCategory="100"
      android:showAsAction="always"
      android:icon="@drawable/ic_action_refresh"
      android:title="Refresh"/>
   <item
      android:id="@+id/action_settings"
      android:title="Settings">
   </item>
</menu>
```

The **MenuInflator** class allows to inflate actions defined in an XML file and adds them to the action bar. MenuInflator can get accessed via the getMenuInflator() method from your activity.

Eg –

```
@Override
   public boolean onCreateOptionsMenu(Menu menu)
{
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.mainmenu, menu);
      return true;
   }
```

****