Introduction to Mobile applications

Mobile apps were originally offered for general productivity and information retrieval, including email, calendar, contacts, the stock market and weather information. However, public demand and the availability of developer tools drove rapid expansion into other categories, such as those handled by desktop application software packages. As with other software, the explosion in number and varietyof apps made discovery a challenge, which in turn led to the creation of a wide range of review, recommendation, and curation sources,including blogs, magazines, and dedicated online app-discovery services.

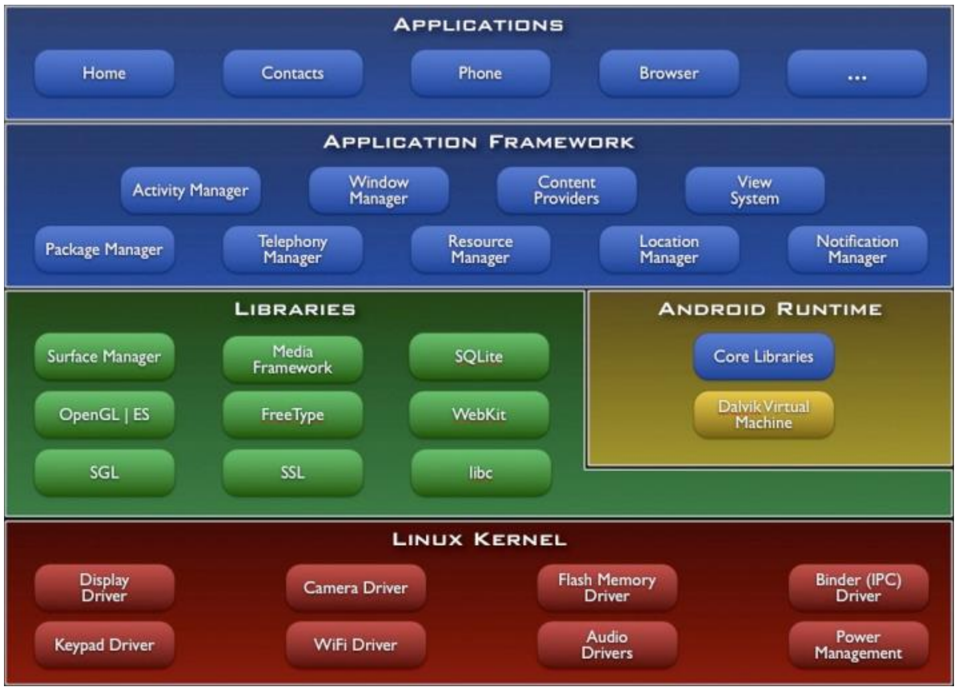
Mobile applications may be classified by numerous methods. A common scheme is to distinguish native, web-based, and hybrid apps. The three biggest app stores are Google Play for Android, App Store for iOS, and Microsoft Store for Windows 10, Windows 10 Mobile, and Xbox One.

Essence of A Mobile Device:
• (Potentially) available to serve everywhere, any time.
• Interwoven into daily life – live, work, play, study
• Represents and intimately "knows" the user – Much more than just a small computer, it represents the user
• Brings in the outside world – sensing, location, communication
• Now the dominant end-user device

Mobile Application Development Challenges
• Competitive, fluid vendor landscape (Apple, Android consortium incl. Amazon, RIM, HP) means apps need to be multi-platform for wide adoption
• No "standard" device (what about iOS, Windows Phone devices?)
• Low bandwidth input (in most cases – what about tablets?)
• Limited screen size (tablets?)
• Unreliability in connectivity and device (network access, power, ambient light, noise, at least for now)
• Integration tradeoffs with cloud and enterprise services

Application Development Support
• 3rd Generation Object-Oriented Languages (iOS – Objective C, Android – Java, Windows Phone – C#)
• Scripting languages (JavaScript, Ruby)
• Cross-platform frameworks – Titanium, RhoMobile, Xamarin, PhoneGap
• C and C++
• Integrated into "frameworks" specifically for mobile application development

Framework Support (e.g. Android)
Blue background: Java Other colors: C/C++



Framework Capabilities and Add-Ons
• Built-In Services:
– GUI, OS services (file I/O, threads, device management), Graphics, Device access (GPS, camera, music, and video players, sensors), Web-services, Networking, XML processing, standard language libraries
• Add-ons:
– Maps
– Database support (SQLite)
– WebKit
IDE Support
• Open IDEs – Eclipse/Android Studio for Android)
• Proprietary (Xcode for iOS, MS Visual Studio)
• Testing tools (test management, unit tests)
• Performance profiling tools
• SCM integration (Git, SVN, CVS)
• Software emulators
• Sensor injection (GPS, accelerometer, others)

Embedded system

An **embedded system** is a computer system—a combination of a computer processor, computer memory, and input/output peripheral devices—that has a dedicated function within a larger mechanical or electronic system. It is *embedded* as part of a complete device often including electrical or electronic hardware and mechanical parts. Because an embedded system typically controls physical operations of the machine that it is embedded within, it often has real-time computing constraints.

Embedded systems control many devices in common use today. In 2009 it was estimated that ninety-eight percent of all microprocessors manufactured were used in embedded systems.
Modern embedded systems are often based on microcontrollers (i.e. microprocessors with integrated memory and peripheral interfaces), but ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also common, especially in more complex systems. In either case, the processor(s) used may be types ranging from general purpose to those specialized in a certain class of computations, or even custom designed for the application at hand.
A common standard class of dedicated processors is the digital signal processor (DSP).
Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.
Embedded systems range in size from portable personal devices such as digital watches and MP3 players to bigger machines like home appliances, industrial assembly lines, robots, transport vehicles, traffic light controllers, and medical imaging systems. Often they constitute subsystems of other machines like avionics in aircraft and spacecraft. Large installations like factories, pipelines and electrical grids rely on multiple embedded systems networked together. Generalized through software customization, embedded systems such as programmable logic controllers frequently comprise their functional units.
Embedded systems range from those low in complexity, with a single microcontroller chip, to very high with multiple units, peripherals and networks, which may reside in equipment racks or across large geographical areas connected via long-distance communications lines.

Applications
Embedded systems are used for safety-critical systems in aerospace and defence industries. Unless connected to wired or wireless networks via on-chip 3G cellular or other methods for IoT monitoring and control purposes, these systems can be isolated from hacking and thus be more secure. For fire safety, the systems can be designed to have a greater ability to handle higher temperatures and continue to operate. In dealing with security, the embedded systems can be self-sufficient and be able to deal with cut electrical and communication systems.
Miniature wireless devices called motes are networked wireless sensors. Wireless sensor networking makes use of miniaturization made possible by advanced IC design to couple full wireless subsystems to sophisticated sensors, enabling people and companies to measure a myriad of things in the physical world and act on this information through monitoring and control systems. These motes are completely self-contained and will typically run off a battery source for years before the batteries need to be changed or charged.
Transportation systems from flight to automobiles increasingly use embedded systems. New airplanes contain advanced avionics such as inertial guidance systems and GPS receivers
that also have considerable safety requirements. Spacecraft rely on avionics systems for trajectory correction.
Various electric motors — brushless DC motors, induction motors and DC motors — use electronic motor controllers. Automobiles, electric vehicles, and hybrid vehicles increasingly use embedded systems to maximize efficiency and reduce pollution. Other automotive safety systems using embedded systems include anti-lock braking system (ABS), Electronic Stability Control (ESC/ESP), traction control (TCS) and automatic four-wheel drive

Characteristics
Embedded systems are designed to do some specific task, rather than be a general-purpose

computer for multiple tasks. Some also have real-time performance constraints that must be met, for reasons such as safety and usability; others may have low or no performance requirements, allowing the system hardware to be simplified to reduce costs.
Embedded systems are not always standalone devices. Many embedded systems consist of small parts within a larger device that serves a more general purpose. For example, the Gibson Robot Guitar features an embedded system for tuning the strings, but the overall purpose of the Robot Guitar is, of course, to play music.
Similarly, an embedded system in an automobile provides the program instructions written for embedded systems are referred to as firmware, and are stored in read-only memory or
flash memory chips. They run with limited computer hardware resources: little memory, small or non-existent keyboard or screen.
Embedded systems range from no user interface at all, in systems dedicated only to one task, to complex graphical user interfaces that resemble modern computer desktop operating systems. Simple embedded devices use buttons, LEDs, graphic or character LCDs (HD44780 LCD for example) with a simple menu system.
More sophisticated devices that use a graphical screen with touch sensing or screen-edge soft keys provide flexibility while minimizing space used: the meaning of the buttons can change with the screen, and selection involves the natural behaviour of pointing at what is desired.
Some systems provide user interface remotely with the help of a serial (e.g. RS-232) or network (e.g. Ethernet) connection.
This approach extends the capabilities of the embedded system, avoids the cost of a display, simplifies BSP and allows designers to build a rich user interface on the PC. A good example of this is the combination of an Embedded HTTP server running on an embedded device (such as an IP camera or a network router). The user interface is displayed in a web browser on a PC connected to the device.
Examples of properties of typical embedded computers, when compared with general-purpose counterparts, are low power consumption, small size, rugged operating ranges, and low per-unit cost. This comes at the price of limited processing resources.
Numerous microcontrollers have been developed for embedded systems use. General-purpose microprocessors are also used in embedded systems, but generally, require more support circuitry than microcontrollers.
PC/104 and PC/104+ are examples of standards for *ready-made* computer boards intended for small, low-volume embedded and ruggedized systems. These are mostly x86-based and often physically small compared to a standard PC, although still quite large compared to most simple (8/16-bit) embedded systems. They may use DOS, Linux, NetBSD, or an embedded real-time operating system (RTOS) such as MicroC/OS-II, QNX or VxWorks.
In certain applications, where small size or power efficiency are not primary concerns, the components used may be compatible with those used in general-purpose x86 personal computers.

ASIC and FPGA SoC solutions

A system on a chip (SoC) contains a complete system - consisting of multiple processors, multipliers, caches, even different types of memory and commonly various peripherals like interfaces for wired or wireless communication on a single chip. Often graphics processing units (GPU) and DSPs are included such chips. SoCs can be implemented as an application-specific integrated circuit (ASIC) or using a field-programmable gate array (FPGA) which typically can be reconfigured.

ASIC implementations are common for very-high-volume embedded systems like mobile phones and smartphones. ASIC or FPGA implementations may be used for not-so-high-volume embedded systems with special needs in kind of signal processing performance, interfaces and reliability, like in avionics.

Peripherals

Embedded systems talk with the outside world via peripherals, such as:

▪ Serial communication interfaces (SCI): RS-232, RS-422, RS-485, etc.
▪ Synchronous Serial Interface: I2C, SPI, SSC and ESSI (Enhanced Synchronous Serial Interface)
▪ Universal Serial Bus (USB)
▪ Media cards (SD cards, CompactFlash, etc.)
▪ Network interface controller: Ethernet, WiFi, etc.
▪ Fieldbuses: CAN bus, LIN-Bus, PROFIBUS, etc.

Embedded debugging may be performed at different levels, depending on the facilities available. Considerations include does it slow down the main application, how close is the debugged system or application to the actual system or application, how expressive are the triggers that can be set for debugging (e.g., inspecting the memory when a particular program counter value is reached), and what can be inspected in the debugging process (such as, only memory, or memory and registers, etc.).

From simplest to most sophisticated debugging techniques and systems be roughly grouped into the following areas:

▪ Interactive resident debugging, using the simple shell provided by the embedded operating system (e.g. Forth and Basic)
▪ Software-only debuggers have the benefit that they do not need any hardware modification but must carefully control what they record in order to conserve time and storage space.[10]
▪ External debugging using logging or serial port output to trace operation using either a monitor in flash or using a debug server like the Remedy Debugger that even works for heterogeneous multicore systems.
▪ An in-circuit debugger (ICD), a hardware device that connects to the microprocessor via a JTAG or Nexus interface.[11] This allows the operation of the microprocessor to be controlled externally, but is typically restricted to specific debugging capabilities in the processor.
▪ An in-circuit emulator (ICE) replaces the microprocessor with a simulated equivalent, providing full control over all aspects of the microprocessor.
▪ A complete emulator provides a simulation of all aspects of the hardware, allowing all of it to be controlled and modified, and allowing debugging on a normal PC. The downsides are expense and slow operation, in some cases up to 100 times slower than the final system.
▪ For SoC designs, the typical approach is to verify and debug the design on an FPGA prototype board. Tools such as Certus are used to insert probes in the FPGA implementation that make signals available for observation. This is used to debug hardware, firmware, and software interactions across multiple FPGAs in an implementation with capabilities like a logic analyser.

Unless restricted to external debugging, the programmer can typically load and run software through the tools, view the code running in the processor, and start or stop its operation. The view of the code may be as High-level programming language, assembly code or mixture of both.

Specific reliability issues may include:

▪ The system cannot safely be shut down for repair, or it is too inaccessible to repair. Examples include space systems, undersea cables, navigational beacons, bore-hole systems, and automobiles.
▪ The system must be kept running for safety reasons. "Limp modes" are less tolerable. Often backups are selected by an operator. Examples include aircraft navigation, reactor control systems, safety-critical chemical factory controls, train signals.
▪ The system will lose large amounts of money when shut down: Telephone switches, factory controls, bridge and elevator controls, funds transfer and market making, automated sales and service.
A variety of techniques are used, sometimes in combination, to recover from errors—both software bugs such as memory leaks, and soft errors in the hardware:
▪ watchdog timer that resets the computer unless the software periodically notifies the watchdog subsystems with redundant spares that can be switched over to software "limp modes" that provide partial function
▪ Designing with a Trusted Computing Base (TCB) architecture ensures a highly secure & reliable system environment

Embedded software architectures

There are several different types of software architecture in common use today.

Simple control loop or Programmed Input-output

In this design, the software simply has a loop which monitors the input device and executes the corresponding subroutine(s) only if there is a new action on the input device. The loop calls subroutines, each of which manages a part of the hardware or software. Hence it is called a simple control loop or programmed Input-output.

Interrupt-controlled system

Some embedded systems are predominantly controlled by interrupts. This means that tasks performed by the system are triggered by different kinds of events; an interrupt could be generated, for example, by a timer in a predefined frequency, or by a serial port controller receiving a byte.
These kinds of systems are used if event handlers need low latency, and the event handlers are short and simple. Usually, these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays.

Cooperative multitasking

Sometimes the interrupt handler will add longer tasks to a queue structure. Later, after the interrupt handler has finished, these tasks are executed by the main loop. This method brings the system close to a multitasking kernel with discrete processes.
A non-preemptive multitasking system is very similar to the simple control loop scheme, except that the loop is hidden in an API.[3][1] The programmer defines a series of tasks, and each task gets its own environment to “run” in. When a task is idle, it calls an idle routine, usually called “pause”, “wait”, “yield”, “nop” (stands for *no operation*), etc.
The advantages and disadvantages are like that of the control loop, except that adding new software is easier, by simply writing a new task, or adding to the queue.

Microkernels and exokernels

microkernel is a logical step up from a real-time OS. The usual arrangement is that the operating system kernel allocates memory and switches the CPU to different threads of execution. User-mode processes implement major functions such as file systems, network interfaces, etc.

In general, microkernels succeed when task switching and inter task communication is fast and fail when they are slow.
Exokernels communicate efficiently by normal subroutine calls. The hardware and all the software in the system are available to and extensible by application programmers

Monolithic kernels

In this case, a relatively large kernel with sophisticated capabilities is adapted to suit an embedded environment. This gives programmers an environment like a desktop operating system like Linux or Microsoft Windows and is therefore very productive for development; on the downside, it requires considerably more hardware resources, is often more expensive, and, because of the complexity of these kernels, can be less predictable and reliable.

Common examples of embedded monolithic kernels are embedded Linux, VxWorks, and Windows CE.

Despite the increased cost in hardware, this type of embedded system is increasing in popularity, especially on the more powerful embedded devices such as wireless routers and GPS navigation systems. Here are some of the reasons:

▪ Ports to common embedded chip sets are available.
▪ They permit re-use of publicly available code for device drivers, web servers, firewalls, and other code.
▪ Development systems can start out with broad feature-sets, and then the distribution can be configured to exclude unneeded functionality, and save the expense of the memory that it would consume.
▪ Many engineers believe that running application code in user mode is more reliable and easier to debug, thus making the development process easier and the code more portable.
▪ Features requiring faster response than can be guaranteed can often be placed in hardware

Business Drivers for Mobile Working

The rapid adoption of mobile services and solutions has been driven by the “perfect storm” of hardware, software, connectivity, business challenges and consumer demand. Here are compelling trends that have combined to accelerate the adoption of mobility.

1. CONSUMERISATION OF IT

Technology has become an integral part of everyday life. Business users are also consumers, and they are using mobile devices in all aspects of their day to- day activity, from social media to online shopping, email, video capture and gaming.
Mobile devices are more user-friendly; interfaces are intuitive, customisable, and consistent. As a result, the consumer experience is having a significant impact on what users expect from their business applications.

2. UBIQIUTOUS CONNECTIVITY

As technology has come to dominate business processes, workers need constant access to data and applications if they are to work effectively. Users have come to demand access to information anywhere, any time and on any device.

Business users also depend on real-time communication to facilitate collaboration, speed up decision making and deliver an excellent customer experience. Mobile devices, flexible working and cloud-enabled computing allow users to operate as a part of a team, wherever they are.

3. USER-CENTRICITY

Convenience, flexibility and performance are key features of mobile computing. Instant-on devices and cloud-based services put the user at the heart of
the process. Desktop virtualisation and bespoke business apps are replacing traditional, feature-heavy business applications.

As users become more mobile, they become increasingly frustrated with traditional IT. Mobility provides the flexibility, efficiency, and convenience that modern users demand from their technology.

4. CLOUD COMPUTING

As businesses continue their journey to the Cloud, processes become more data-driven and device-independent. Public, private or hybrid Cloud infrastructures are being used to deliver business essentials such as email.

5. COST OF OWNERSHIP

Competition amongst mobile device and platform providers continues to drive quality up and prices down. Screen resolution, processing power, battery life and security continue to improve and with most devices in the workplace becoming user-owned, the fixed costs of provisioning and replacing devices is being eliminated from the business, allowing IT to spend Op Ex on device and application management.

Mobile Marketing Overview

Mobile Marketing is a next generation trend of marketing products and services. Today, the strategy of mobile marketing is being practiced by most of the traders and businessmen across the globe through which they communicate or engage with their potential audiences/customers in an interactive and relevant manner. The companies create short, but interactive messages (to promote their business) specifically to deliver on mobile devices.

Why Mobile Marketing?

Mobiles have gained unprecedented importance in our lives, today. People around the world choose mobile devices as their preferred medium to connect with other people, gather information or even do business. As a result, many businesses are actively devising new mobile marketing strategies to reach out to their audience. In the present world, mobile marketing is a common technique that almost every company, irrespective of the business it is into, is pursuing mobile marketing campaign.

Whether it’s an E-commerce giant such as Amazon or a manufacturing juggernaut such as General Motors, everyone is following a “mobile first” approach, when it comes to creating a marketing strategy for their businesses.

Businesses are also aware that the use and influence of mobile phones today have gone up since the days of the simple text & voice messages. People order meals from a restaurant, buy & Read a book/magazine/news, and find a childhood friend all with a single tap on them smartphones. Keeping this fact in mind, this tutorial describes various kinds of mobile marketing campaign, adopted by the companies, important of them are −

- SMS Campaign
- Mobile Website Campaign
- Mobile Apps Campaign
- Mobile Advertising Campaign
- Mobile Social Media Marketing Campaign
- Mobile Email-Marketing Campaign
- M-Commerce Campaign

Mobile Marketing Strategies

People use their mobile phones to not only remain in contact, but also for reading the review of the products, knowing about the new products in the market, and of course online purchasing. In such a condition, developing a masterly designed mobile marketing strategy is significant for your business.

What is Mobile Marketing Strategy?

Before you begin developing a mobile marketing strategy, you should figure out how it will fit with your other marketing plans. It will help you to find out what is important to your business. Is product branding a current goal? Or is customer acquisition and lead generation a priority task? Maybe it’s none of these and more about social media engagement and viral marketing. Whatever it is, you must synergize your marketing strategy with the current priorities of your company.

Research for Information

Market research related to your business is the first step. Here are some tips on what kinds of research you should be doing −

- Research how mobile marketing is done in your industry; you can join online forums or partner with someone.
- Gather data relevant to your product or service such as case studies, research analysis reports, whitepapers, etc.; and

• Do a comparative analysis by surfing their websites, press releases, online campaign, etc.

Identify Your Target Audience

Identifying your target audience and their choices will help you to take better decisions and develop successful marketing campaign. Following are the three important steps that you need to follow while identifying your target audience −

• Make a detailed list of potential customers and give them actual names and identities.
• Conduct online surveys, emails of customers etc. to understand what questions they have.
• Create customer personas by visualizing specific attributes such as their age, profession, task they perform etc.

Define the Value of Your Offerings

One of the most significant steps before designing a mobile campaign is to have the answers ready for the following questions −

• How is your product/service beneficial to your customers?
• What are the additional benefits you are offering in comparison to your competitors?
• How is it going to fulfil their needs?
• How will it fit their budget?

Understanding the value or **unique selling proposition** (USP) of your product or service helps you to engage your customers better right from the time you first communicate with them to the time they become repeat customers. Once you define the value offering in your marketing strategy, the task of making new customers and maintaining the existing ones become much easier.

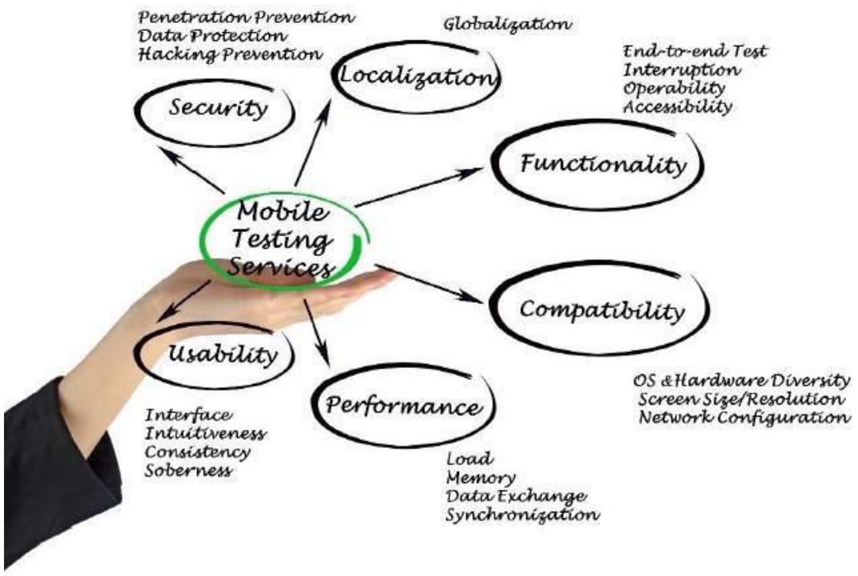
Outline Your Goals & Objectives

What you want to achieve with your marketing efforts is one of the most important questions. It determines the budget you allocate towards marketing and the channels you choose to market your product and services. You must define it clearly what you want to accomplish. Is it more sales? Or is it more brand awareness? The outlining of your goals and objectives is the cornerstone of your marketing strategy.

Mobile Technology & Reach

Mobile technology has grown leaps and bounds over the last few decades. The journey from the clunky wireless phone to sleek smartphone has been peppered by several amazing innovations and discoveries.

With the 3G mobiles available today, users can do a lot more with their smartphones than just sending a voicemail or SMS. They can browse the web, check the weather, read a book, prepare a to-do-list, carry their favourite music around, find their way around a new city with GPRS and do much more.



Mobile Marketing Technology & Reach

In such a scenario, every businessperson willing to practice the mobile marketing technique has to be aware about the following top 5 ways in which mobile technologies have changed the way users interact with mobile devices –

Anywhere anytime access

Unlike laptops and desktops, tablet and mobile devices are easy to carry around. Users can access the internet in their mobile devices at anytime and anywhere and it has decreased use of the laptops.

Mobile Apps are easier than websites

Companies that have an online customer base for instance e-commerce portals have noticed that the sales and subscriptions they get from their apps are higher in comparison to that of their websites. This means, online purchasing from the mobile phones is easier.

Advertisements should be personalized

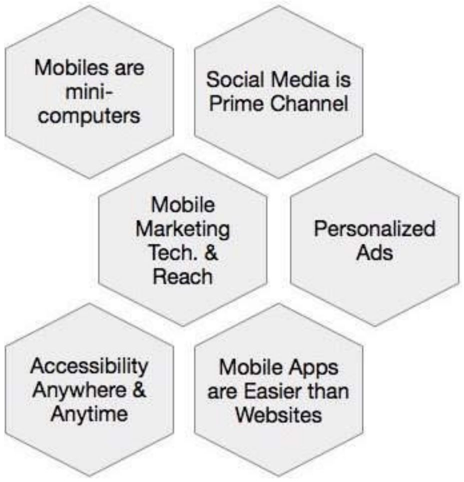
Most of the users feel that the mobile is more personal device than say a laptop or a desktop. Therefore, they expect that the content they receive on their mobile phones must be personalized as well.

Social media is a prime channel

Whether your customers are business professionals, students, homemakers, teenagers etc., commonly, they spend more than 3 hours a week on social media channels, such as Twitter, Facebook etc. Surprisingly, most of these customers access these channels on their smartphones.

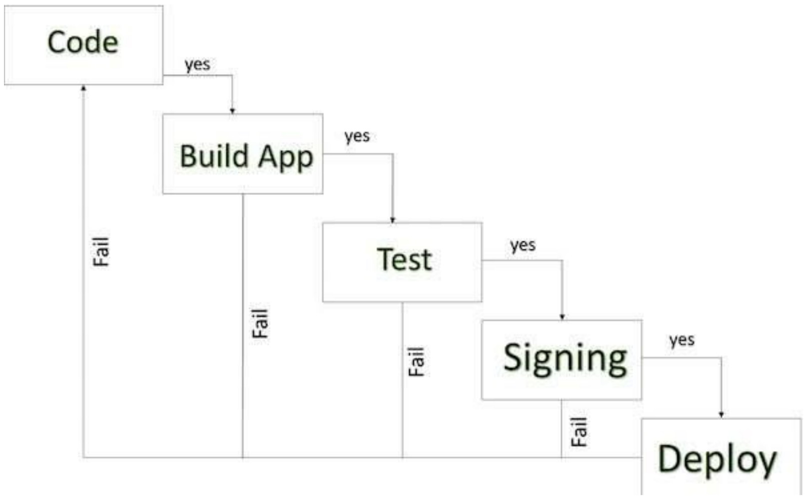
Mobiles are turning into minicomputers

Slowly but surely technicians and developers are packing the mobile phones with computer-like features. With bigger screens, faster performance, optimum storage capacity, longer battery life, and a ton of productivity booster applications. The evolution of phones from a simple calling device to multi-tasking-pocket-size computers has revolutionized the world.


Mobiles are mini-computers
Social Media is Prime Channel
Mobile Marketing Tech. & Reach
Personalized Ads
Accessibility Anywhere & Anytime
Mobile Apps are Easier than Websites

<u>**Publishing Android Application**</u>

Android application publishing is a process that makes your Android applications available to users. Infect, publishing is the last phase of the Android application development process.



Android development life cycle

Once you developed and fully tested your Android Application, you can start selling or distributing free using Google Play (A famous Android marketplace). You can also release your applications by sending them directly to users or by letting users download them from your own website.

You can check a detailed publishing process at Android official website, but this tutorial will take you through simple steps to launch your application on Google Play. Here is a simplified check list which will help you in launching your Android application –

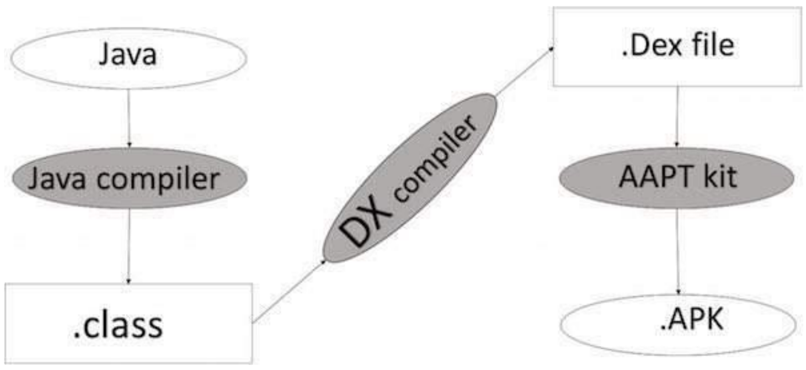
Step Activity

1 Regression Testing Before you publish your application, you need to make sure that its meeting the basic quality expectations for all Android apps, on all the devices that you are targeting. So, perform all the required testing on different devices including phone and tablets.

2 Application Rating When you will publish your application at Google Play, you will have to specify a content rating for your app, which informs Google Play users of its

maturity level. Currently available ratings are (a) Everyone (b) Low maturity (c) Medium maturity (d) High maturity.
3 Targeted Regions Google Play lets you control what countries and territories where your application will be sold. Accordingly you must take care of setting up time zone, localization or any other specific requirement as per the targeted region.
4 Application Size Currently, the maximum size for an APK published on Google Play is 50 MB. If your app exceeds that size, or if you want to offer a secondary download, you can use APK Expansion Files, which Google Play will host for free on its server infrastructure and automatically handle the download to devices.
5 SDK and Screen Compatibility It is important to make sure that your app is designed to run properly on the Android platform versions and device screen sizes that you want to target.
6 Application Pricing Deciding whether you app will be free or paid is important because, on Google Play, free app's must remain free. If you want to sell your application then you will have to specify its price in different currencies.
7 Promotional Content It is a good marketing practice to supply a variety of high-quality graphic assets to showcase your app or brand. After you publish, these appear on your product details page, in store listings and search results, and elsewhere.
8 Build and Upload release-ready APK The release-ready APK is what you will upload to the Developer Console and distribute to users. You can check complete detail on how to create a release-ready version of your app: Preparing for Release.
9 Finalize Application Detail Google Play gives you a variety of ways to promote your app and engage with users on your product details page, from colourful graphics, screen shots, and videos to localized descriptions, release details, and links to your other apps. So, you can decorate your application page and provide as much as clear crisp detail you can provide.

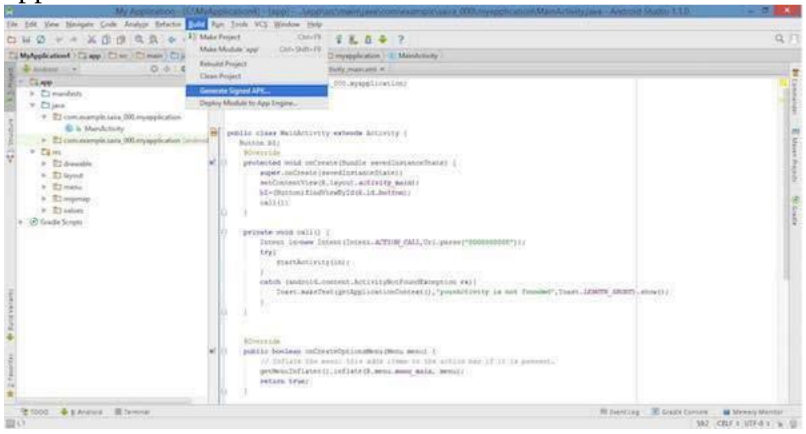
<u>Export Android Application Process</u>



Apk development process

Before exporting the apps, you must some of tools
• **Dx tools**(Dalvik executable tools): It going to convert **.class file** to **.dex file**. it has useful for memory optimization and reduce the boot-up speed time
• **AAPT**(Android assistance packaging tool):it has useful to convert **.Dex file** to**.Apk**
• **APK**(Android packaging kit): The final stage of deployment process is called as .apk.
You will need to export your application as an APK (Android Package) file before you upload it

Google Play marketplace.
To export an application, just open that application project in Android studio and select **Build → Generate Signed APK** from your Android studio and follow the simple steps to export your application –

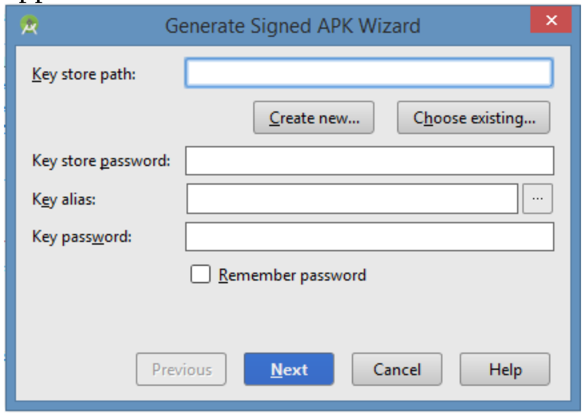


```
public class MainActivity extends Activity {
   Button b1;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      b1=(Button)findViewById(R.id.button);
      call();
   }

   private void call() {
      Intent in=new Intent(Intent.ACTION_CALL,Uri.parse("0000000000"));
      try{
         startActivity(in);
      }
      catch (android.content.ActivityNotFoundException ex){
         Toast.makeText(getApplicationContext(),"yourActivity is not founded",Toast.LENGTH_SHORT).show();
      }
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is present.
      getMenuInflater().inflate(R.menu.menu_main, menu);
      return true;
   }
```

Next select, **Generate Signed APK** option as shown in the above screen shot and then click it so that you get following screen where you will choose **Create new keystore** to store your application.


Generate Signed APK Wizard

Key store path:

Create new... Choose existing...

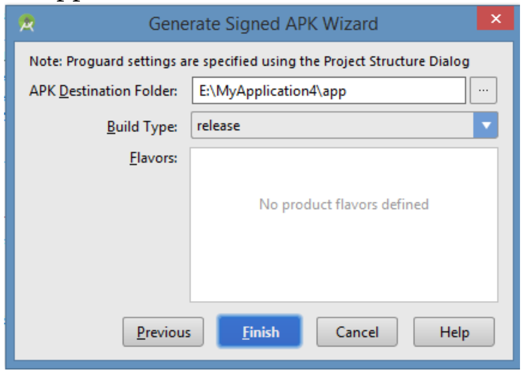
Key store password:

Key alias:

Key password:

☐ Remember password

Enter your key store path,key store password,key alias and key password to protect your application and click on **Next** button once again. It will display following screen to let you create an application –


Generate Signed APK Wizard

Note: Proguard settings are specified using the Project Structure Dialog

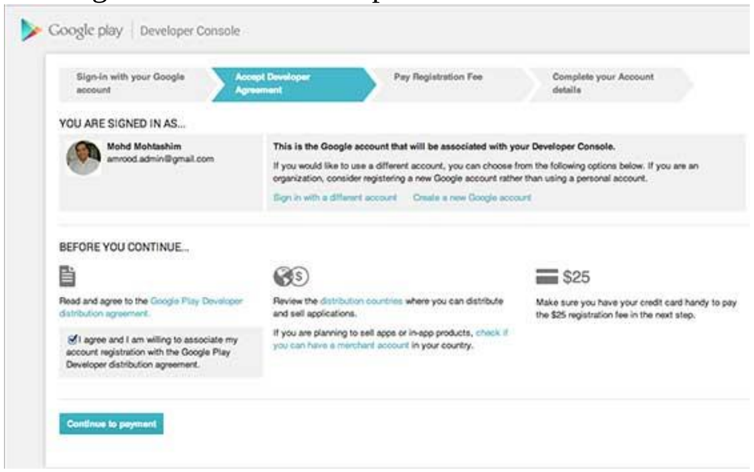
APK Destination Folder: E:\MyApplication4\app

Build Type: release

Flavors: No product flavors defined

Once you filled up all the information, like app destination, build type and flavours click **finish** button While creating an application it will show as below

Google Play Registration
The most important step is to register with Google Play using Google Play Marketplace . You can use your existing google ID if you have any otherwise you can create a new Google ID and then register with the marketplace. You will have following screen to accept terms and condition.


Google play | Developer Console
Sign-in with your Google account
Accept Developer Agreement
Pay Registration Fee
Complete your Account details
Continue to payment

YOU ARE SIGNED IN AS...

Mohd Mohtashim
amrood.admin@gmail.com

This is the Google account that will be associated with your Developer Console.

If you would like to use a different account, you can choose from the following options below. If you are an organization, consider registering a new Google account rather than using a personal account.

Sign in with a different account Create a new Google account

BEFORE YOU CONTINUE...

Read and agree to the Google Play Developer distribution agreement.

[✓] I agree and I am willing to associate my account registration with the Google Play Developer distribution agreement.

Review the distribution countries where you can distribute and sell applications.

If you are planning to sell apps or in-app products, check if you can have a merchant account in your country.

$25

Make sure you have your credit card handy to pay the $25 registration fee in the next step.

You can use **Continue to payment** button to proceed to make a payment of $25 as a registration fee and finally to complete your account detail.
Once you are a registered user at Google Play, you can upload **release ready APK** for your application and finally you will complete application detail using application detail page as mentioned in step 9 of the above-mentioned checklist.

Signing Your App Manually
You do not need Android Studio to sign your app. You can sign your app from the command line using standard tools from the Android SDK and the JDK. To sign an app in release mode from the command line −

- Generate a private key using keytool

```
$ keytool -genkey -v -keystore my-release-key.keystore
-alias alias_name -keyalg RSA -keysize 2048 -validity 10000
```

- Compile your app in release mode to obtain an unsigned APK
- Sign your app with your private key using jarsigner

```
$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore
my_application.apk alias_name
```

- Verify that your APK is signed. For example −

```
$ jarsigner -verify -verbose -certs my_application.apk
```

- Align the final APK package using zipalign.

```
$ zipalign -v 4 your_project_name-unaligned.apk your_project_name.apk
```

Requirement Gathering Techniques
Techniques describe how tasks are performed under specific circumstances. A task may have none or one or more related techniques. A technique should be related to at least one task.
The following are some of the well-known requirements gathering techniques −

Brainstorming
Brainstorming is used in requirement gathering to get as many ideas as possible from group of people. Generally used to identify possible solutions to problems, and clarify details of opportunities.

Document Analysis

Reviewing the documentation of an existing system can help when creating AS–IS process document, as well as driving gap analysis for scoping of migration projects. In an ideal world, we would even be reviewing the requirements that drove creation of the existing system – a starting point for documenting current requirements. Nuggets of information are often buried in existing documents that help us ask questions as part of validating requirement completeness.

Focus Group

A focus group is a gathering of people who are representative of the users or customers of a product to get feedback. The feedback can be gathered about needs/opportunities/ problems to identify requirements or can be gathered to validate and refine already elicited requirements. This form of market research is distinct from brainstorming in that it is a managed process with specific participants.

Interface analysis

Interfaces for a software product can be human or machine. Integration with external systems and devices is just another interface. User centric design approaches are very effective at making sure that we create usable software. Interface analysis – reviewing the touch points with other external systems are important to make sure we don't overlook requirements that aren't immediately visible to users.

Interview

Interviews of stakeholders and users are critical to creating the great software. Without understanding the goals and expectations of the users and stakeholders, we are very unlikely to satisfy them. We also must recognize the perspective of each interviewee, so that, we can properly weigh and address their inputs. Listening is the skill that helps a great analyst to get more value from an interview than an average analyst.

Observation

By observing users, an analyst can identify a process flow, steps, pain points and opportunities for improvement. Observations can be passive or active (asking questions while observing). Passive observation is better for getting feedback on a prototype (to refine requirements), were active observation is more effective at getting an understanding of an existing business process. Either approach can be used.

Prototyping

Prototyping is a relatively modern technique for gathering requirements. In this approach, you gather preliminary requirements that you use to build an initial version of the solution - a prototype. You show this to the client, who then gives you additional requirements. You change the application and cycle around with the client again. This repetitive process continues until the product meets the critical mass of business needs or for an agreed number of iterations.

Requirement Workshops

Workshops can be very effective for gathering requirements. More structured than a brainstorming session, involved parties collaborate to document requirements. One way to capture the collaboration is with creation of domain-model artifacts (like static diagrams, activity diagrams). A workshop will be more effective with two analysts than with one.

Reverse Engineering

When a migration project does not have access to sufficient documentation of the existing system, reverse engineering will identify what the system does. It will not identify what the system should do and will not identify when the system does the wrong thing.

Survey/Questionnaire

When collecting information from many people – too many to interview with budget and time constraints – a survey or questionnaire can be used. The survey can force users to select from choices, rate something (“Agree Strongly, agree…”), or have open ended questions allowing free-form responses. Survey design is hard – questions can bias the respondents.