IV YEAR - VII SEMESTER

CS8079 – HUMAN COMPUTER INTERACTION (R 2017)

UNIT IV MOBILE HCI

Mobile Ecosystem: Platforms, Application frameworks- Types of Mobile Applications: Widgets, Applications, Games- Mobile Information Architecture, Mobile 2.0, Mobile Design: Elements of Mobile Design, Tools. - Case Studies

4.0 BRIEF HISTORY OF MOBILE

The Brick Era calls

- It was Portable!
- More expensive than payphones
- Enormous battery
- Stakeholders: Stockbrokers, sales people, ...
- After a while, more cellular radio towers and… it got (a little bit) smaller

The Candy Bar Era calls SMS

- 2G network : GSM, CDMA, TDMA, iDEN
- More cellular towers
  - less power needed
  - much smaller
- Better voice quality
- Added SMS
- Everyone wanted to have a mobile phone
  - – economic prosperity in EU, USA, and JP

The Feature Phone Era calls SMS & MMS music & photos

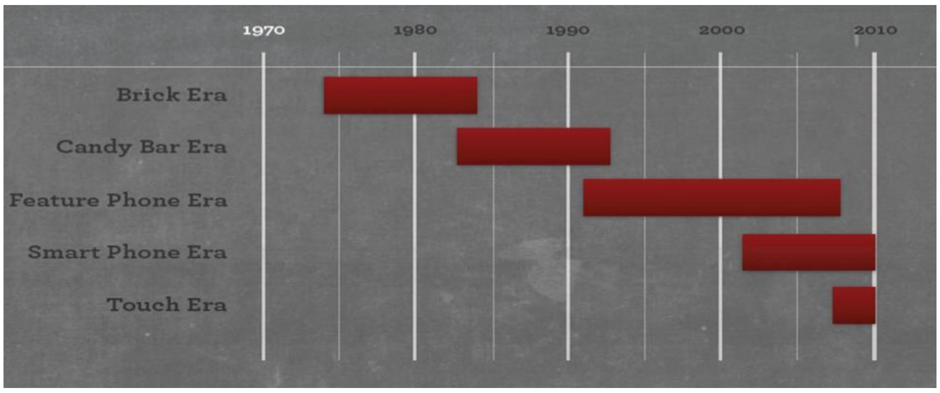
- 2.5G network: GPRS
- Camera
- MMS
- Data-capable devices
- Internet on mobile (very poor)
  - high prices
  - poor marketing
  - inconsistent rendering

The Smartphone Era calls SMS & MMS music & photos

- 3G, HSDPA, WI-FI
- Like a feature phone, but simulating a PC
- Its own OS (es. Symbian)
- Larger screens, stylus
- The Mobile Platform becomes key
- (push) email as primary driver

The Touch Era calls SMS & MMS music & photos APPS
(NOT a phone - NOT a computer)
3G, 4G

- Accelerometers
- GPS/Location-based
- User-centered design



4.1 MOBILE ECOSYSTEM

Mobile is an entirely unique ecosystem and, like the Internet, it is made up of many different parts that must all work seamlessly together. However, with mobile technology, the parts are different, and because you can use mobile devices to access the Internet, that means that not only do you need to understand the facets of the Internet, but you also need to understand the mobile ecosystem. Think of the mobile ecosystem as a system of layers, as shown in Figure 4.1. Each layer is reliant on the others to create a seamless, end-to-end experience. Although not every piece of the puzzle is included in every mobile product and service, for the majority of the time, they seem to add complexity to our work, regardless of whether we expressly put them there.

Services
Applications
Application frameworks
Operating Systems
Platforms
Devices
Aggregators
Networks
Operators

Figure 4.1. The layers of the mobile ecosystem

Although the mobile ecosystem consists of many different components, probably the most recognizable and important one is the mobile phone. All phones sold today fall into one of three categories: feature phones, smart phones, or touch phones.

- **Feature Phones:** The vast majority of cell phone users in the US have feature phones. These are the basic flip phones that come free or at a low-cost with carrier contracts and pre-paid plans. Features phones get their name from the various features that come with the devices. These phones generally have camera, a handful of applications, and rudimentary web browsers. Prior to feature phones, cell phones only made calls and sent and received text messages.
- **Smart phones:** They makeup a much smaller portion of the US market than feature phones, but smartphones are also very popular devices, especially among attorneys. The most recognizable smartphone is the Blackberry. In addition to all the capabilities of feature phones, smartphones typically run more applications and an operating system, have a larger screen size, and utilize a QWERTY keyboard input (standard keyboard format).
- **Touch phones:** Since the introduction of the iPhone in 2007, the fastest growing category of phones in the US market have been touch phones. Touchphones can be thought of as the next generation of smartphones - they have larger screens, more robust web browsers, and more powerful applications. Touchphone users are also the mobile web's power users. A recent UK study showed that 93% of iPhone owners use their device to access news and information on the mobile web.
- **Other mobile devices** - iPads & tablets are also entering the mobile space and should be watched as their market share increases, especially considering that higher end models have built-in wifi capabilities.

4.1.1 Operators

- Operators are also referred as Mobile Network Operators (MNOs); mobile service providers, wireless carriers, or simply carriers; mobile phone operators; or cellular companies
- Essentially make the entire mobile ecosystem work (Gate keepers)
- Operator's role in the ecosystem is to create and maintain a specific set of wireless services over a reliable cellular network
- to create and maintain wireless services over a reliable cellular network
- Operators operate wireless networks

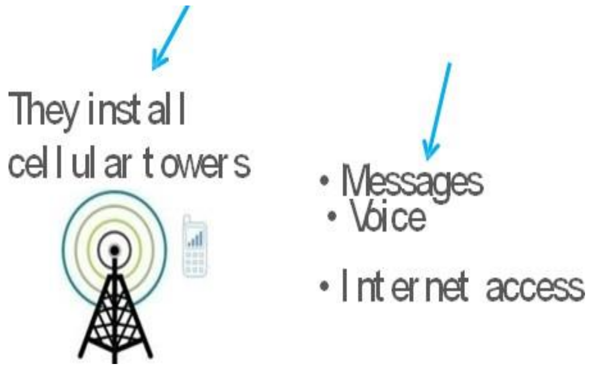
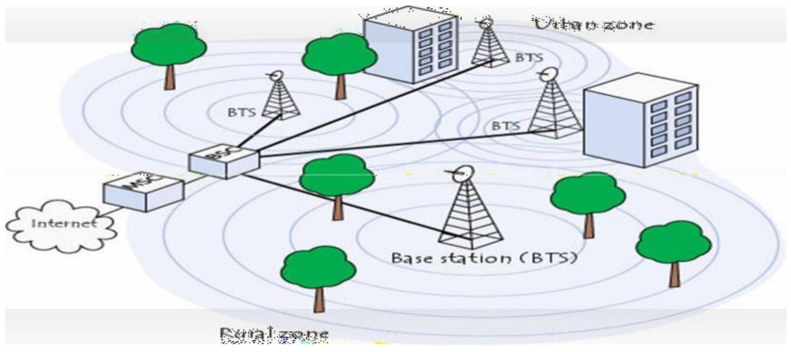


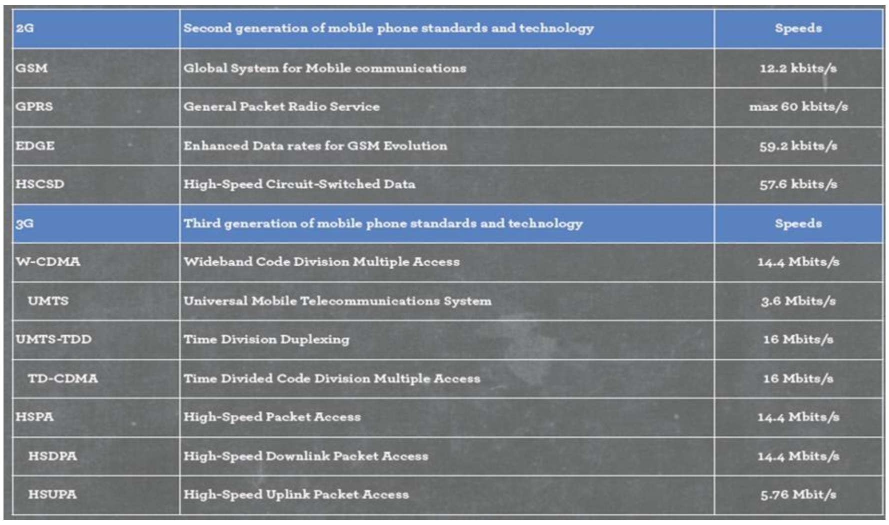
Table 2-1. World's largest mobile operators

Rank	Operator	Markets	Technology	Subscribers (in millions)
1.	China Mobile	China (including Hong Kong) and Pakistan	GSM, GPRS, EDGE, TD-SCDMA	436.12
2.	Vodafone	United Kingdom, Germany, Italy, France, Spain, Romania, Greece, Portugal, Netherlands, Czech Republic, Hungary, Ireland, Albania, Malta, Northern Cyprus, Faroe Islands, India, United States, South Africa, Australia, New Zealand, Turkey, Egypt, Ghana, Fiji, Lesotho, and Mozambique	GSM, GPRS, EDGE, UMTS, HSDPA	260.5
3.	Telefónica	Spain, Argentina, Brazil, Chile, Colombia, Ecuador, El Salvador, Guatemala, Mexico, Nicaragua, Panama, Peru, Uruguay, Venezuela, Ireland, Germany, United Kingdom, Czech Republic, Morocco, and Slovakia	CDMA, CDMA2000 1x, EV-DO, GSM, GPRS, EDGE, UMTS, HSDPA	188.9
4.	América Móvil	United States, Argentina, Chile, Colombia, Paraguay, Uruguay, Mexico, Puerto Rico, Ecuador, Jamaica, Peru, Brazil, Dominican Republic, Guatemala, Honduras, Nicaragua, Ecuador, and El Salvador	CDMA, CDMA2000 1x, EV-DO, GSM, GPRS, EDGE, UMTS, HSDPA	172.5
5.	Telenor	Norway, Sweden, Denmark, Hungary, Montenegro, Serbia, Russia, Ukraine, Thailand, Bangladesh, Pakistan, and Malaysia	GSM, GPRS, EDGE, UMTS, HSDPA	143.0
6.	China Unicom	China	GSM, GPRS	127.6
7.	T-Mobile	Germany, United States, United Kingdom, Poland, Czech Republic, Netherlands, Hungary, Austria, Croatia, Slovakia, Macedonia, Montenegro, Puerto Rico, and U.S. Virgin Islands	GSM, GPRS, EDGE, UMTS, HSDPA	126.6
8.	TeliaSonera	Norway, Sweden, Denmark, Finland, Estonia, Latvia, Lithuania, Spain, and Central Asia	GSM, GPRS, EDGE, UMTS, HSDPA	115.0

4.1.2 NETWORKS

Mobile networks communicate through electromagnetic radio waves with a cell site base station, the antennas of which are usually mounted on a tower,

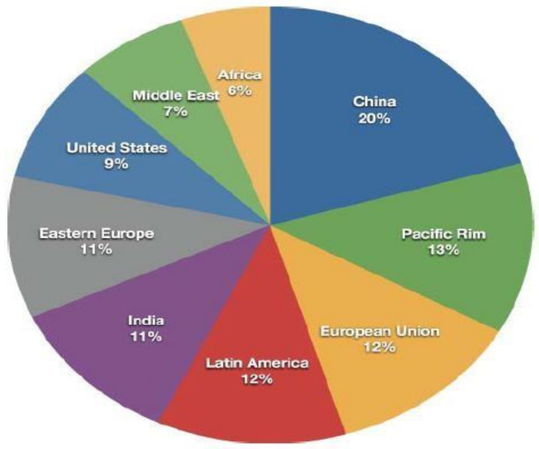
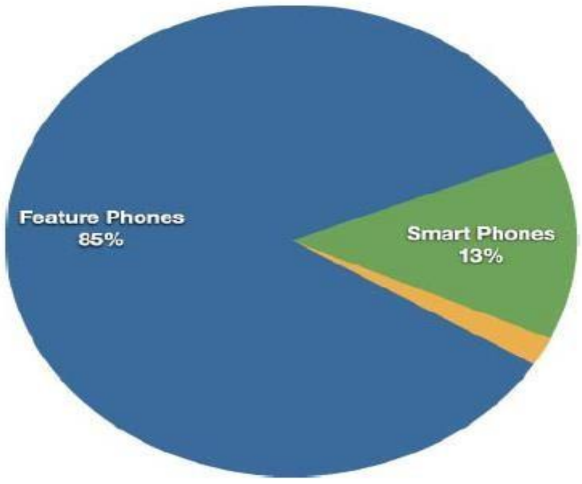




2G	Second generation of mobile phone standards and technology	Speeds
GSM	Global System for Mobile communications	12.2 kbits/s
GPRS	General Packet Radio Service	max 60 kbits/s
EDGE	Enhanced Data rates for GSM Evolution	59.2 kbits/s
HSCSD	High-Speed Circuit-Switched Data	57.6 kbits/s
3G	Third generation of mobile phone standards and technology	Speeds
W-CDMA	Wideband Code Division Multiple Access	14.4 Mbits/s
UMTS	Universal Mobile Telecommunications System	3.6 Mbits/s
UMTS-TDD	Time Division Duplexing	16 Mbits/s
TD-CDMA	Time Divided Code Division Multiple Access	16 Mbits/s
HSPA	High-Speed Packet Access	14.4 Mbits/s
HSDPA	High-Speed Downlink Packet Access	14.4 Mbits/s
HSUPA	High-Speed Uplink Packet Access	5.76 Mbit/s

4.1.3 DEVICES

handsets or terminals in industry , But also other devices such as tablets, ebook readers…

	
Mobile Devices around the world	Breakdown of devices

- **Feature Phones:** The vast majority of cell phone users in the US have feature phones. These are the basic flip phones that come free or at a low-cost with carrier contracts and pre-paid plans. Features phones get their name from the various features that come with the devices. These phones generally have camera, a handful of applications, and rudimentary web browsers. Prior to feature phones, cell phones only made calls and sent and received text messages.
- **Smart phones:** They makeup a much smaller portion of the US market than feature phones, but smartphones are also very popular devices, especially among attorneys. The most recognizable smartphone is the Blackberry. In addition to all the capabilities of feature phones, smartphones typically run more applications and an operating system, have a larger screen size, and utilize a QWERTY keyboard input (standard keyboard format).
- **Touch phones:** Since the introduction of the iPhone in 2007, the fastest growing category of phones in the US market have been touch phones. Touchphones can be thought of as the next generation of smartphones - they have larger screens, more robust web browsers, and more powerful applications. Touchphone users are also the mobile web's power users. A recent UK study showed that 93% of iPhone owners use their device to access news and information on the mobile web.

Other mobile devices - iPads & tablets are also entering the mobile space and should be watched as their market share increases, especially considering that higher end models have built-in wifi capabilities.

4.1.4 PLATFORMS

A mobile platform's primary duty is to provide access to the devices. To run software and services on each of these devices, you need a platform, or a core programming language in which all of your software is written. Like all software platforms, these are split into three categories:

1. Open Source: free to use and modify
    - Android
2. Proprietary: by device makers
    - iPhone, BlackBerry, Palm
3. Licensed: sold to device makers
    - JavaME, BREW, Windows Mobile

1) Licensed:

Licensed platforms are sold to device makers for nonexclusive distribution on devices. The goal is to create a common platform of development Application Programming Interfaces (APIs) that work similarly across multiple devices with the least possible effort required to adapt for device differences, although this is hardly reality. Following are the licensed platforms:
Java Micro Edition (Java ME): Formerly known as J2ME, Java ME is by far the most predominant software platform of any kind in the mobile ecosystem. It is a licensed subset of the Java platform and provides a collection of Java APIs for the development of software for resource constrained devices such as phones.
Binary Runtime Environment for Wireless (BREW): BREW is a licensed platform created by Qualcomm for mobile devices, mostly for the U.S. market. It is an interface-independent platform that runs a variety of application frameworks, such as C/C++, Java, and Flash Lite.
Windows Mobile: Windows Mobile is a licensable and compact version of the Windows operating system, combined with a suite of basic applications for mobile devices that is based on the Microsoft Win32 API.
LiMo: LiMo is a Linux-based mobile platform created by the LiMo Foundation. Although Linux is open source, LiMo is a licensed mobile platform used for mobile devices. LiMo includes SDKs for creating Java, native, or mobile web applications using the WebKit browser framework.

2) Proprietary

Proprietary platforms are designed and developed by device makers for use on their devices. They are not available for use by competing device makers. These include:
Palm: Palm uses three different proprietary platforms. Their first and most recognizable is the Palm OS platform based on the C/C++ programming language; this was initially developed for their Palm Pilot line, but is now used in low-end smartphones such as the Centro line. As Palm moved into higher-end smartphones, they started using the Windows Mobile-based platform for devices like the Treo line. The most recent platform is called webOS, is based on the WebKit browser framework, and is used in the Prē line.

BlackBerry: Research in Motion maintains their own proprietary Java-based platform, used exclusively by their BlackBerry devices.
Iphone: Apple uses a proprietary version of Mac OS X as a platform for their iPhone and iPod touch line of devices, which is based on Unix.

3) Open Source

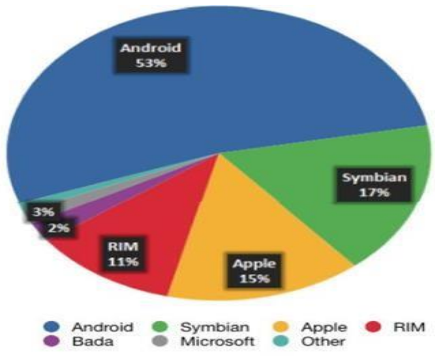
Open source platforms are mobile platforms that are freely available for users to down load, alter, and edit. Open source mobile platforms are newer and slightly controversial, but they are increasingly gaining traction with device makers and developers. Android is one of these platforms. It is developed by the Open Handset Alliance, which is spear-headed by Google. The Alliance seeks to develop an open source mobile platform based on the Java programming language.

Operating Systems

It used to be that if a mobile device ran an operating system, it was most likely considered a smartphone. But as technology gets smaller, a broader set of devices supports operating systems. Operating systems often have core services or toolkits that enable applications to talk to each other and share data or services. Mobile devices without operating systems typically run "walled" applications that do not talk to anything else. Although not all phones have operating systems, the following are some of the most common:
Symbian: Symbian OS is a open source operating system designed for mobile devices, with associated libraries, user interface frameworks, and reference implementations of common tools.
Windows Mobile: Windows Mobile is the mobile operating system that runs on top of the Windows Mobile platform.
Palm OS: Palm OS is the operating system used in Palm's lower-end Centro line of mobile phones.
Linux: The open source Linux is being increasingly used as an operating system to power smartphones, including Motorola's RAZR2.
Mac OS X: A specialized version of Mac OS X is the operating system used in Apple's iPhone and iPod touch.
Android: Android runs its own open source operating system, which can be customized by operators and device manufacturers.
You might notice that many of these operating systems share the same names as the platforms on which they run. Mobile operating systems are often bundled with the platform they are designed to run on.

4.1.5 OPERATING SYSTEM

OS have core services or toolkits that enable apps to talk to each other and share data or services.OSs are common in Smart Phones, but rare in Feature phones.



Smart Phones by OS

- Mobile devices without operating systems typically run “walled” applications that do not talk to anything else.
- Although not all phones have operating systems, the following are some of the most common:
- *Symbian :* Symbian OS is a open source operating system designed for mobile devices, with associated libraries, user interface frameworks, and reference implementations of common tools.
- *Windows Mobile : M*obile operating system that runs on top of the Windows

  Mobile platform.
- *Palm OS : OS* used in Palm’s lower-end Centro line of mobile phones.
- *Linux :* The open source Linux is being increasingly used as an operating system to power smartphones, including Motorola’s RAZR2.
- *Mac OS X:*A specialized version of Mac OS X is the operating system used in Apple’s iPhone and iPod touch.
- *Android : runs its own open source operating system, which can be customized by operators and device manufacturers.*

4.1.6 APPLICATION FRAMEWORKS

Often, the first layer the developer can access is the application framework or API released by one of the companies mentioned already. The first layer that you have any control over is the choice of application framework. Application frameworks often run on top of operating systems, sharing core services such as communications, messaging, graphics, location, security, authentication, and many others.

Java: Applications written in the Java ME framework can often be deployed across the majority of Java-based devices, but given the diversity of device screen size and processor power, cross-device deployment can be a challenge. Most Java applications are purchased and distributed through the operator, but they can also be downloaded and installed via cable or over the air.

S60: The S60 platform, formerly known as Series 60, is the application platform for devices that run the Symbian OS. S60 is often associated with Nokia devices—Nokia owns the platform—but it also runs on several non-Nokia devices. S60 is an open source frame-work. S60 applications can be created in Java, the Symbian C++ framework, or even Flash Lite.

BREW: Applications written in the BREW application framework can be deployed across the majority of BREW-based devices, with slightly less cross-device adaption than other frameworks.

However BREW applications must go through a costly and timely certification process and can be distributed only through an operator.

Flash Lite: Adobe Flash Lite is an application framework that uses the Flash Lite and ActionScript frameworks to create vector-based applications. Flash Lite applications can be run within the Flash Lite Player, which is available in a handful of devices around the world. Flash Lite is a promising and powerful platform, but there has been some difficulty getting it on devices. A distribution service for applications written in Flash Lite is long overdue.

Windows Mobile: Applications written using the Win32 API can be deployed across the majority of Windows Mobile-based devices. Like Java, Windows Mobile applications can be downloaded and installed over the air or loaded via a cable-connected computer.

Cocoa Touch: Cocoa Touch is the API used to create native applications for the iPhone and iPod touch. Cocoa Touch applications must be submitted and certified by Apple before being included in the App Store. Once in the App Store, applications can be purchased, downloaded, and installed over the air or via a cable-connected computer.

Android SDK: The Android SDK allows developers to create native applications for any device that runs the Android platform. By using the Android SDK, developers can write applications in C/C++ or use a Java virtual machine included in the OS that allows the creation of applications with Java, which is more common in the mobile ecosystem.

Web Runtimes (WRTs): Nokia, Opera, and Yahoo! provide various Web Runtimes, or WRTs. These are meant to be miniframeworks, based on web standards, to create mobile widgets. Both Opera's and Nokia's WRTs meet the W3C-recommended specifications for mobile widgets. Although WRTs are very interesting and provide access to some device functions using mobile web principles, but have found them to be more complex than just creating a simple mobile web app, as they force the developer to code within an SDK rather than just code a simple web app. And based on the number of mobile web apps written for the iPhone versus the number written for other, more full-featured WRTs. Nonetheless, it is a move in the right direction.

WebKit: With Palm's introduction of webOS, a mobile platform based on WebKit, and given its predominance as a mobile browser included in mobile platforms like the iPhone, Android, and S60, and that the vast majority of mobile web apps are written specifically for WebKit. WebKit is a browser technology, so applications can be created simply by using web technologies such as HTML, CSS, and JavaScript. WebKit also supports a number of recommended standards not yet implemented in many desktop browsers. Applications can be run and tested in any WebKit browser, desktop, or mobile device.

The Web: The Web is the only application framework that works across virtually all devices and all platforms. Although innovation and usage of the Web as an application framework in mobile has been lacking for many years, increased demand to offer products and services outside of operator control, together with a desire to support more devices in shorter development cycles, has made the Web one of the most rapidly growing mobile application platforms to date.

4.1.7 APPLICATIONS

Definition: In the realm of technology, this usually refers to a computer program that runs on a website (Google Apps), a small computing device (iPad App) or a cell phone (Android App).

- Apps live between the device and the user
- They must fit with their usage context
- They must know the specific device attributes and capabilities
- FRAGMENTATION PROBLEM

4.1.7 SERVICES

Services are “everything the user is trying to do”
They are often available at different levels:

- Application
- Application Framework
- OS

Example services may include:

- the Internet
- sending a text message
- being able to get a location

All of these layers must be passed through before you get to the content

7th Mass MEDIA - MOBILE

1)	Printing Press
2)	Recordings
3)	Cinema
4)	Radio
5)	Television
6)	Internet
7)	MOBILE

4.2 TYPES OF MOBILE APPLICATIONS

The mobile medium type is the type of application framework or mobile technology that presents content or information to the user. It is a technical approach regarding which type of medium to use; this decision is determined by the impact it will have on the user experience. The technical capabilities and capacity of the publisher also factor into which approach to take. Earlier the common mobile platforms was discussed in terms of how they factor in the larger mobile ecosystem. Now let us look deeper into each of these platforms from a more tactical perspective, unpacking them, so to speak, to see what is inside. Figure 4.2 illustrates the spectrum of mobile media; it starts with the basic text- based experiences and moves on to the more immersive experiences.

Figure 4.2. Multiple mobile application medium types

MOBILE APPLICATION MEDIUM TYPES

1. **SMS**
2. **mobile websites**
3. **mobile web widgets**
4. **mobile web applications**
5. **native applications, and**
6. **Games**

4.2.1 SMS:

The most basic mobile application you can create is an SMS application. Although it might seem odd to consider text messages applications, they are nonetheless a designed experience. Given the ubiquity of devices that support SMS, these applications can be useful tools when integrated with other mobile application types.
Typically, the user sends a single keyword to a five-digit short code in order to return information or a link to premium content. For example, sending the keyword “freebie” to a hypothetical short code “12345” might return a text message with a coupon code that could be redeemed at a retail location, or it could include a link to a free ringtone. SMS applications can be both “free,” meaning that there is no additional charge beyond the text message fees an operator charges, or “premium,” meaning that you are charged an additional fee in exchange for access to premium content. The most common uses of SMS applications are mobile content, such ringtones and images, and to interact with actual goods and services. Some vending machines can dispense beverages when you send them an SMS; SMS messages can also be used to purchase time at a parking meter or pay lot. A great example of how SMS adds incredible value would be Twitter, where users can receive SMS alerts from their friends and post to their timeline from any mobile device.

Pros : The pros of SMS applications include:

- They work on any mobile device nearly instantaneously.
- They're useful for sending timely alerts to the user.
- They can be incorporated into any web or mobile application.
- They can be simple to set up and manage.

Cons : The cons of SMS applications include:

- They're limited to 160 characters.
- They provide a limited text-based experience.
- They can be very expensive.

4.2.2 Mobile Websites:

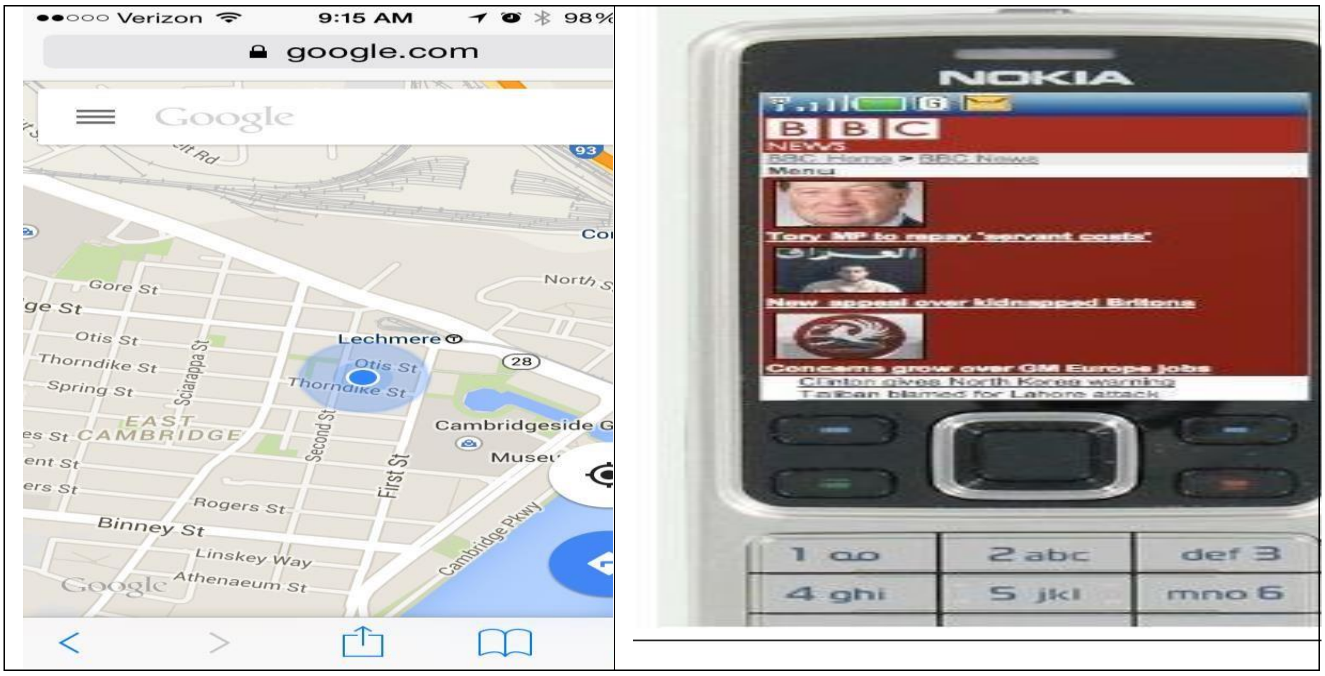


Figure 4.3. An example of a mobile website

As you might expect, a mobile website is a website designed specifically for mobile devices, not to be confused with viewing a site made for desktop browsers on a mobile browser. Mobile websites are characterized by their simple "drill-down" architecture, or the simple presentation of navigation links that take you to a page a level deeper, as shown in Figure 4.3.

Mobile websites often have a simple design and are typically informational in nature, offering few—if any—of the interactive elements you might expect from a desktop site. Mobile websites have made up the majority of what was consider the mobile web for the past decade, starting with the early WML-based sites (not much more than a list of links) and moving to today's websites, with a richer experience that more closely resembles the visual aesthetic users have come to expect with web content.
Though mobile websites are fairly easy to create, they fail to display consistently across multiple mobile browsers—a trait common to all mobile web mediums. The mobile web has been gradually increasing in usage over the years in most major markets, but the limited experience offered little incentive to the user. Many compare the mobile web to a 10-year-old version of the Web: slow, expensive to use, and not much to look at.
As better mobile browsers started being introduced to device platforms like the iPhone and Android, the quality of mobile websites began to improve dramatically, and with it, usage improved. For example, in just one year, the U.S. market went from being just barely in the top five consumers of the mobile web to number one, largely due to the impact of the iPhone alone.

Pros : The pros of mobile websites are:

- They are easy to create, maintain, and publish.
- They can use all the same tools and techniques you might already use for desktop sites.
- Nearly all mobile devices can view mobile websites.

Cons : The cons of mobile websites are:

- They can be difficult to support across multiple devices.
- They offer users a limited experience.
- Most mobile websites are simply desktop content reformatted for mobile devices.
- They can load pages slowly, due to network latency.

4.2.3 Mobile Web WIDGETS

Largely in response to the poor experience provided by the mobile web over the years, there has been a growing movement to establish mobile widget frameworks and platforms. For years the mobile web user experience was severely underutilized and failed to gain traction in the market, so several operators, device makers, and publishers began creating widget platforms (as shown in figure 4.4) to counter the mobile web's weaknesses. Initially user saw mobile web widgets as another attempt by the mobile industry to hype a technology that no one wants. So in order to define a mobile web widget: A component of a user interface that operates in a particular way. The ever-trusty Wikipedia defines a web widget this way: A portable chunk of code that can be installed and executed within any separate HTML- based web page by an end user without requiring additional compilation. Between these two definitions is a better answer: A mobile web widget is a standalone chunk of HTML-based code that is executed by the end user in a particular way.

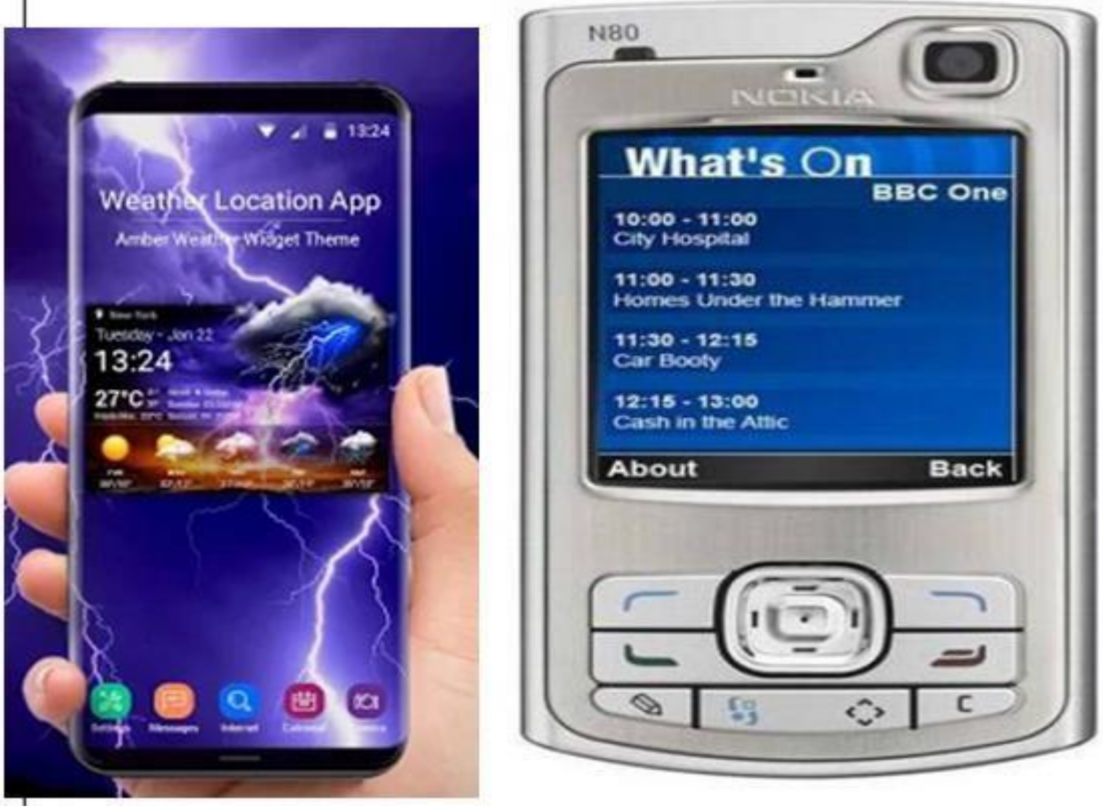


Figure 4.4 : An example mobile web widget

Basically, mobile web widgets are small web applications that can't run by themselves; they need to be executed on top of something else. One reason for all the confusion around what is a mobile web widget is that this definition can also encompass any web application that runs in a browser. Opera Widgets, Nokia Web RunTime (WRT), Yahoo! Blueprint, and Adobe Flash Lite are all examples of widget platforms that work on a number of mobile handsets. Using a basic knowledge of HTML (or vector graphics in the case of Flash), user can create compelling user experiences that tap into device features and, in many cases, can run while the device is offline. Widgets, however, are not to be confused with the utility application context, a user experience designed around short, task-based operations.

Pros: The pros of mobile web widgets are:

- They are easy to create, using basic HTML, CSS, and JavaScript knowledge.
- They can be simple to deploy across multiple handsets.
- They offer an improved user experience and a richer design, tapping into device features and offline use.

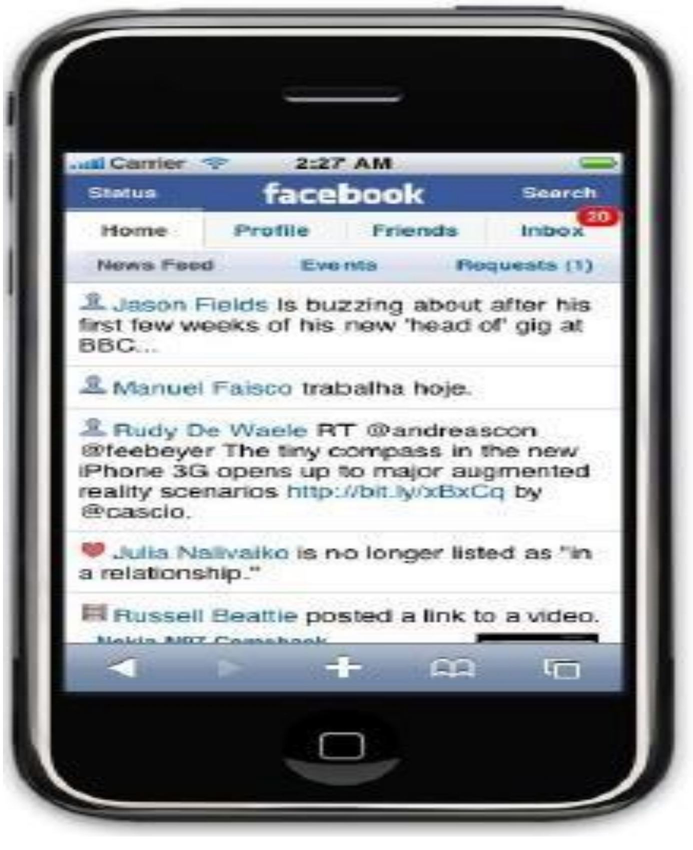
Cons : The cons of mobile web widgets are:

- They typically require a compatible widget platform to be installed on the device.
- They cannot run in any mobile web browser.
- They require learning additional proprietary, non-web-standard techniques.

4.2.4 Mobile web APPLICATIONS

Mobile web applications are mobile applications that do not need to be installed or compiled on the target device. Using XHTML, CSS, and JavaScript, they are able to provide an application-like experience to the end user while running in any mobile web browser. By "application-like" experience, mean that they do not use the drill-down or page metaphors in which a click equals a refresh of the content in view. Web applications allow users to interact with content in real time, where a click or touch performs an action within the current view.

The history of how mobile web applications came to be so commonplace is interesting, and is one that can give an understanding of how future mobile trends can be assessed and understood. Shortly after the explosion of Web 2.0, web applications like Facebook, Flickr, and Google Reader hit desktop browsers, and there was discussion of how to bring those same web applications to mobile devices. The Web 2.0 movement brought user-centered design principles to the desktop web, and those same principles were sorely needed in the mobile web space as well.



Jason Fields is buzzing about after his first few weeks of his new 'head of' gig at BBC...

Manuel Faisco trabalha hoje.

Rudy De Waele RT @andreascon @feebeyer The tiny compass in the new iPhone 3G opens up to major augmented reality scenarios http://bit.ly/xBxCq by @cascio.

Julia Nalivaiko is no longer listed as "in a relationship."

Russell Beattie posted a link to a video.

Figure 4.5 : The Facebook mobile web app

The challenge, as always, was device fragmentation. The mobile browsers were years behind the desktop browsers, making it nearly impossible for a mobile device to render a comparable experience. While XHTML support had become fairly commonplace across devices, the rendering of CSS2 was wildly inconsistent, and support for Java-Script, necessary or simple DHTML, and Ajax was completely nonexistent.

To make matters worse, the perceived market demand for mobile web applications was not seen as a priority with many operators and device makers. It

was the classic chicken-or-the-egg scenario. What had to come first, market demand to drive browser innovation or optimized content to drive the market?
With the introduction of the first iPhone, we saw a cataclysmic change across the board. Using WebKit, the iPhone could render web applications not optimized for mobile devices as perfectly usable, including DHTML- and Ajax-powered content. Developers quickly got on board, creating mobile web applications optimized mostly for the iPhone (as shown in figure 4.5). The combination of a high-profile device with an incredibly powerful mobile web browser and a quickly increasing catalog of nicely optimized experiences created the perfect storm the community had been waiting for.
Usage of the mobile web exploded with not just users of the iPhone, but users of other handsets, too. Because web applications being created for the iPhone were based on web standards, they actually worked reasonably well on other devices. Operators and device makers saw that consumers wanted not just the mobile web on their handsets, but the regular Web, too. In less than a year, we saw a strong unilateral move by all operators and devices makers to put better mobile web browsers in their phones that could leverage this new application medium. We have not seen such rapid innovation in mobile devices since the inclusion of cameras.
The downside, of course, like all things mobile-web-related, is that not all devices support the capability to render mobile web applications consistently. However, we do see a prevalent trend that the majority of usage of the mobile web is coming from the devices with better browsers, in some markets by a factor of 7:1. So although creating a mobile web application might not reach all devices, it will reach the devices that create the majority of traffic.
Pros : The pros of mobile web applications are:

- They are easy to create, using basic HTML, CSS, and JavaScript knowledge.
- They are simple to deploy across multiple handsets.
- They offer a better user experience and a rich design, tapping into device features and offline use.
- Content is accessible on any mobile web browser.

Cons : The cons of mobile web applications are:

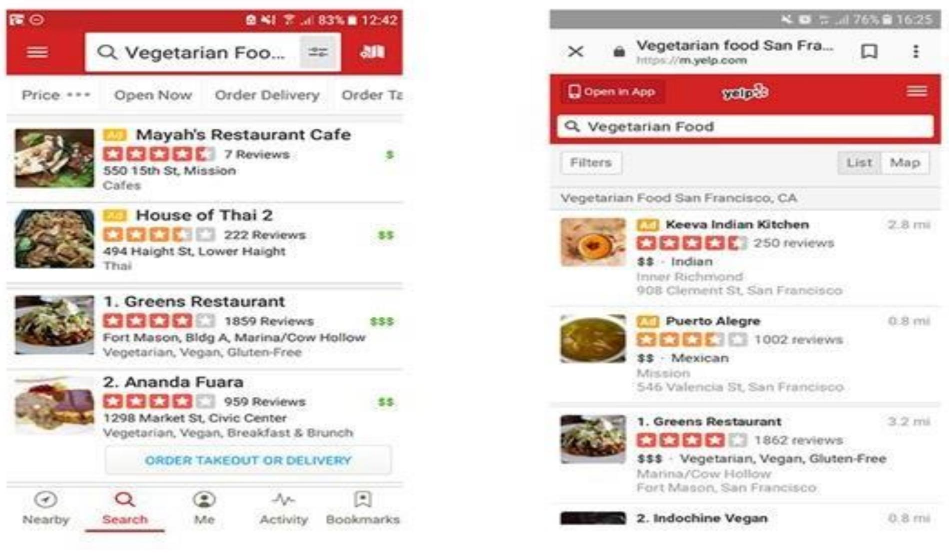
- The optimal experience might not be available on all handsets.
- They can be challenging (but not impossible) to support across multiple devices.
- They don't always support native application features, like offline mode, location lookup, file system access, camera, and so on.

4.2.5 NATIVE APPLICATIONS

- called "platform applications,"
- have to be developed and compiled for each mobile platform.
- native or platform applications are built specifically for devices that run the platform
- most common of all platforms is Java ME (formerly J2ME)
- In addition to Java, other smartphone programming languages include versions of C, C++, and Objective-C

- Because platform applications sit on top of the platform layer, they can tap into the majority of the device features, working online or offline, accessing the location and the filesystem
- the majority (70 %) of native applications in use today could be created with a little bit of XHTML, CSS, and JavaScript

Yelp native app vs. Yelp.com web app



Difference

Web apps

- need an active internet connection in order to run,
- will update themselves

Mobile Native apps

- may work offline.
- faster and more efficient, but they do require the user to regularly download updates.

Pros

The pros of native applications include:

- They offer a best-in-class user experience, offering a rich design and tapping into device features and offline use.
- They are relatively simple to develop for a single platform.
- You can charge for applications.

Cons

The cons of native applications include:

- They cannot be easily ported to other mobile platforms.
- Developing, testing, and supporting multiple device platforms is incredibly costly.
- They require certification and distribution from a third party that you have no control over.
- They require you to share revenue with the one or more third parties.

4.2.6 GAMES

The final mobile medium is games, the most popular of all media available to mobile devices. Technically games are really just native applications that use the similar platform SDKs to create immersive experiences (as shown in figure 4.6). But this is treated differently from native applications for two reasons: they cannot be easily duplicated with web technologies, and porting them to multiple mobile platforms is a bit easier than typical platform-based applications.

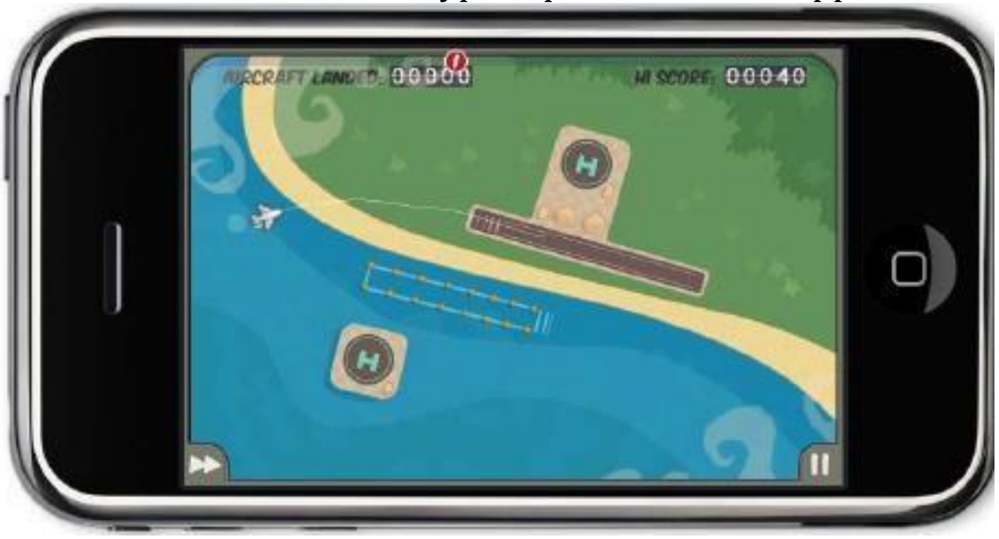


Figure 4.6 : An example game for the iPhone

Although user can do many things with a powerful mobile web browser, creating an immersive gaming experience is not one of them—at least not yet. Seeing as how these types of gaming experiences appear on the desktop using standard web technologies, but is believed we are still a few years out from seeing them on mobile devices. Adobe's Flash and the SVG (scalable vector graphics) standard are the only way to do it on the Web now, and will likely be how it is done on mobile devices in the future, the primary obstacle being the performance of the device in dealing with vector graphics.

The reason games are relatively easy to port ("relatively" being the key word), is that the bulk of the gaming experience is in the graphics and actually uses very little of the device APIs. The game mechanics are the only thing that needs to adapted to the various platforms. Like in console gaming, there are a great number of mobile game porting shops that can quickly take a game written in one language and port it to another.

These differences, are what make mobile games stand apart from all other application genres—their capability to be unique and difficult to duplicate in another application type, though the game itself is relatively easy to port.

Pros: The pros of game applications are:

- They provide a simple and easy way to create an immersive experience.
- They can be ported to multiple devices relatively easily.

Cons : The cons of game applications are:

- They can be costly to develop as an original game title.
- They cannot easily be ported to the mobile web.

Table 6-1. Mobile application media matrix

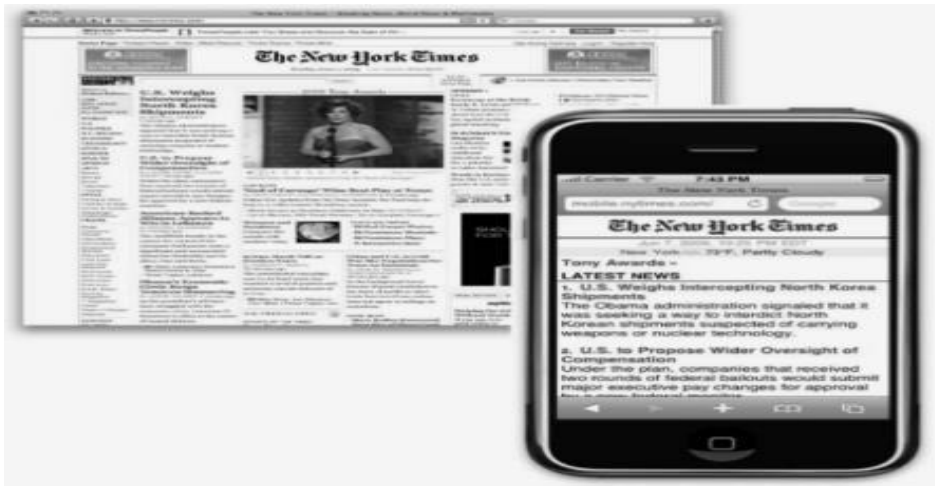
	Device support	Complexity	User experience	Language	Offline support	Device features
SMS	All	Simple	Limited	N/A	No	None
Mobile websites	All	Simple	Limited	HTML	No	None
Mobile web widgets	Some	Medium	Great	HTML	Limited	Limited
Mobile web applications	Some	Medium	Great	HTML, CSS, JavaScript	Limited	Limited
Native applications	All	Complex	Excellent	Various	Yes	Yes
Games	All	Complex	Excellent	Various	Yes	Yes

4.3 MOBILE INFORMATION ARCHITECTURE

It defines not just how your information will be structured, but also how people will interact with it.

Although information architecture has become a common discipline in the web industry, unfortunately, the mobile industry has only a handful of specialized mobile information architects.

For example, if we look at the front page of http://www.nytimes.com as seen from a desktop web browser compared to how it may render in a mobile browser (as shown in figure 4.7), we see a content-heavy site that works well on the desktop, and is designed to present the maximum amount of information above the "fold" or where the screen cuts off the content. However, in the mobile browser, the text is far too small to be useful.



1. U.S. Weighs Intercepting North Korea Shipments
The Obama administration signaled that it was seeking a way to interdict North Korean shipments suspected of carrying weapons or nuclear technology.
2. U.S. to Propose Wider Oversight of Compensation
Under the plan, companies that received two rounds of federal bailouts would submit major executive pay changes for approval

Figure 4.7: Comparing the New York Times website in desktop &mobile browsers

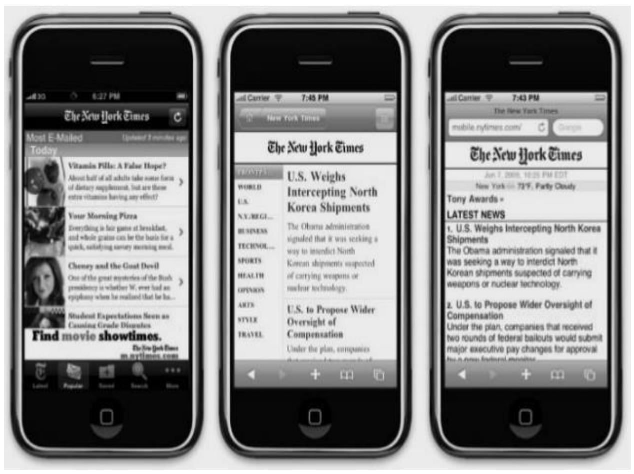


Figure 4.8 : The many mobile experiences of the New York Times

The role of a mobile information architect would be to interpret this content to the mobile context. Do you use the same structure, or sections? Do you present the same information above the fold? If so, how should that be prioritized? How does the user navigate to other areas? Do you use the same visual and interaction paradigms, or invent new ones? And if you do start to invent new paradigms, will you lose the visual characteristics of what users expect?
These are only some of the questions asked when starting to create a mobile information architecture. As you can see in figure above there are several different ways that the New York Times has been interpreted for the mobile context. Also it is needed to design our mobile information architecture to address the mobile context. Given that many devices can detect user current location, which is one of the most immediate types of context, how does the New York Times application address the user's context? For example, as a publication that serves both New York City and a larger global audience, if user is not in New York, should user still see the local New York headlines? Or should user see the headlines based on current location? This is shown in figure 4.8.

Keeping It Simple
When thinking about your mobile information architecture, you want to keep it as simple as possible.

Support your defined goals: If something doesn't support the defined goals, lose it. Go back to the user goals and needs, and identify the tasks that map to them. Find those needs and fill them.
Ask yourself: what need does my application fill? What are people trying to do here? Once it is understand that, it is a simple process of reverse-engineering the path from where they want to be to where they are starting. Cut out everything else.

Clear, simple labels: Good trigger labels, the words used to describe each link or action, are crucial in Mobile. Words like "products" or "services" aren't good trigger labels. They don't tell anything about that content or what can expect. Users have a much higher threshold of pain when clicking about on a desktop

site or application, hunting and pecking for tasty morsels. Mobile performs short, to-the-point, get-it-quick, and get-out types of tasks. What is convenient on the desktop might be a deal breaker on mobile. Keep all labels short and descriptive, and never try to be clever with the words used to evoke action.
The worst sin is to introduce branding or marketing into information architecture; this will just serve to confuse and distract users. Executives love to use the words they use internally to external communications on websites and applications, but these words have no meaning outside of your company walls. Don't try to differentiate product offering by what it is called. Create something unique by creating a usable and intuitive experience based on focusing on what users need and using the same language they use to describe those needs.
Based on what web design is, should use simple, direct terms for navigating around pages rather than overly clever terms. That latter typically result in confused visitors who struggle to find the content they are looking for. When that happens, they will go elsewhere to look for the information they want. So, if the these same mistakes is applied to a constrained device like mobile, then it ends up adding confusion to the user experience at a higher magnitude than the Web.

Site Map:
The first deliverable we use to define mobile information architecture is the site map. Site maps are a classic information architecture deliverable. They visually represent the relationship of content to other content and provide a map for how the user will travel through the informational space, as shown in Figure 4.9. Mobile site maps aren't that dissimilar from site maps we might use on the Web. But there are a few tips specific to mobile that we want to consider.

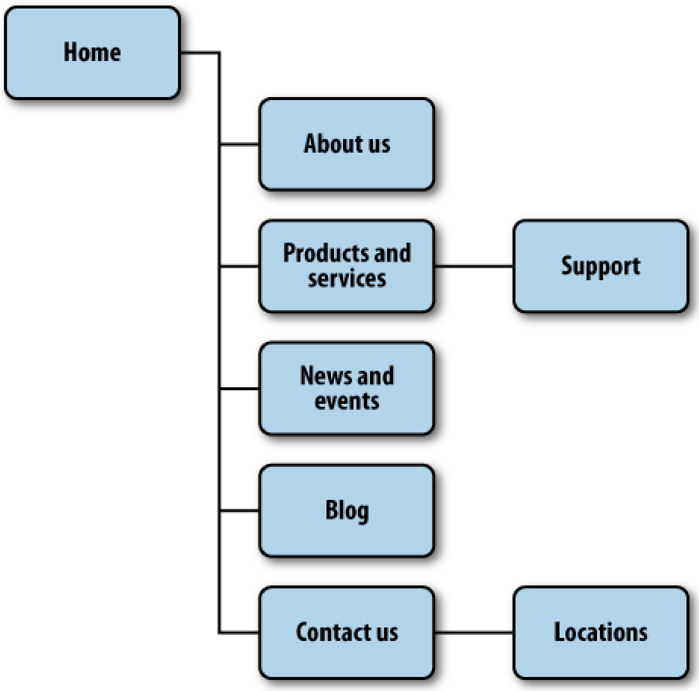


Figure 4.9 : An example mobile site map

Limit opportunities for mistakes:
Now think of your own website. How many primary navigation areas do you have? Seven? Eight? Ten? Fifteen? What risk is there to the users for making a wrong choice? If they go down the wrong path, they can immediately click back to where

they started and go down another path, eliminating the wrong choices to find the right ones. The risks for making the wrong choice are minor. In Figure 4.10, there is a poorly designed mobile information architecture that too closely mimics its desktop cousin; it was not designed with the mobile user in mind.
But in mobile, this assumption cannot be maked. In the mobile context, tasks are short and users have limited time to perform them. And with mobile websites, it can't be assumed that the users have access to a reliable broadband connection that allows them to quickly go back to the previous page. In addition, the users more often than not have to pay for each page view in data charges. So not only do they pay cash for viewing the wrong page by mistake, they pay to again download the page they started from. Therefore, it is to limit users' options, those forks in the road, to five or less. Anything more, and have to introduce far too much risk that the user will make a mistake and head off in the wrong direction.

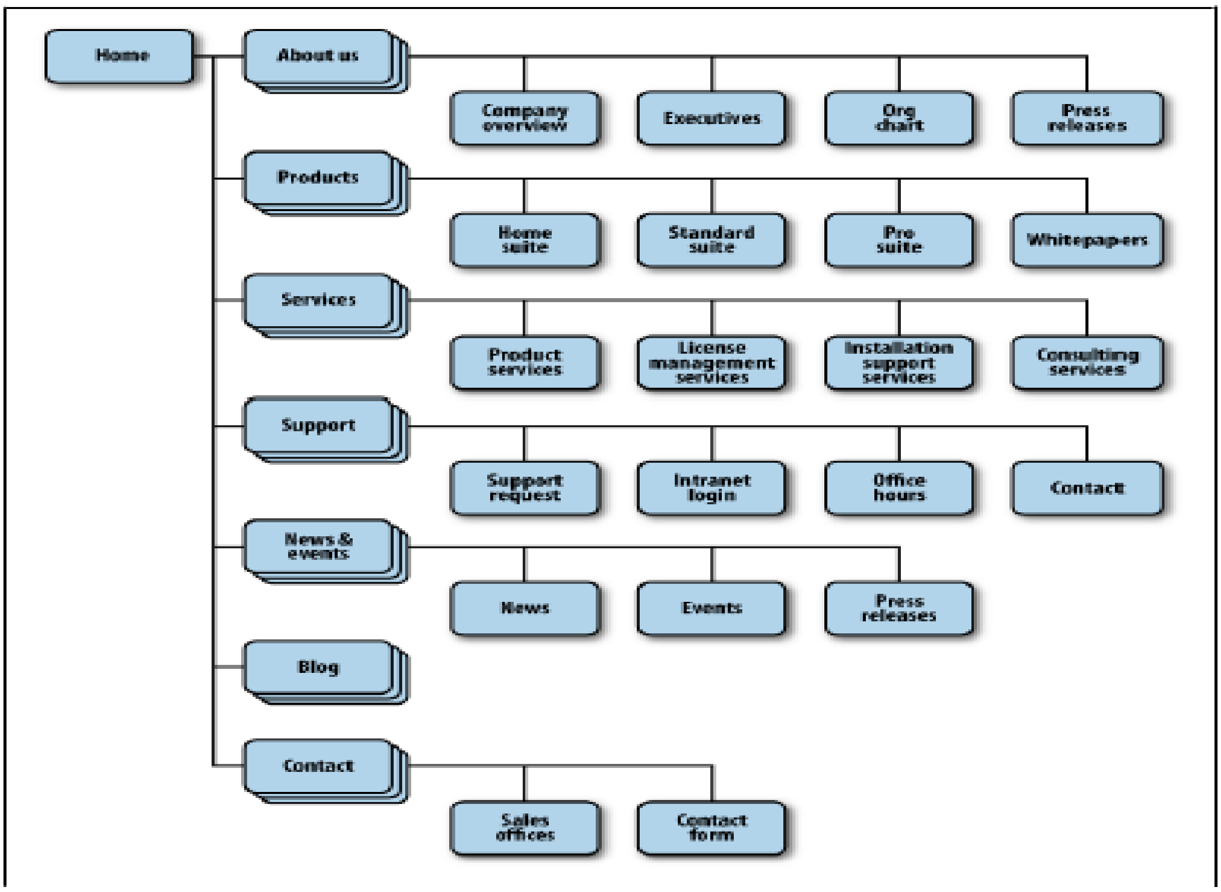


Figure 4.10: An example of a bad mobile information architecture that was designed with desktop users in mind rather than mobile users

Confirm the path by teasing content:
After the users have selected a path, it isn't always clear whether they are getting to where they need to be. Information-heavy sites and applications often employ nested or drill-down architectures, forcing the user to select category after category to get to their target. To reduce risking the user's time and money, have to make sure that enough information is present for the user to wade through the information architecture successfully. On the Web, these risks are taken very lightly, but not with mobile. This is done by teasing content within each category that is, providing at least one content item per category.

The challenge with ringtone sites is there are a lot of items, grouped by artist, album, genre, and so on. The user starts with a goal like "I want a new ringtone" and finds an item that suits his taste within a catalog of tens of thousands of items.
In order to make sense of a vast inventory of content, we have to group, subgroup, and sometimes even +subgroup again, creating a drill-down path for the user to browse. Though on paper this might seem like a decent solution, once you populate an application with content, the dreaded "Page 1 of 157" appears. What user would ever sit there with a mobile device and page through 157 pages of ringtones? What user would page through five pages of content?

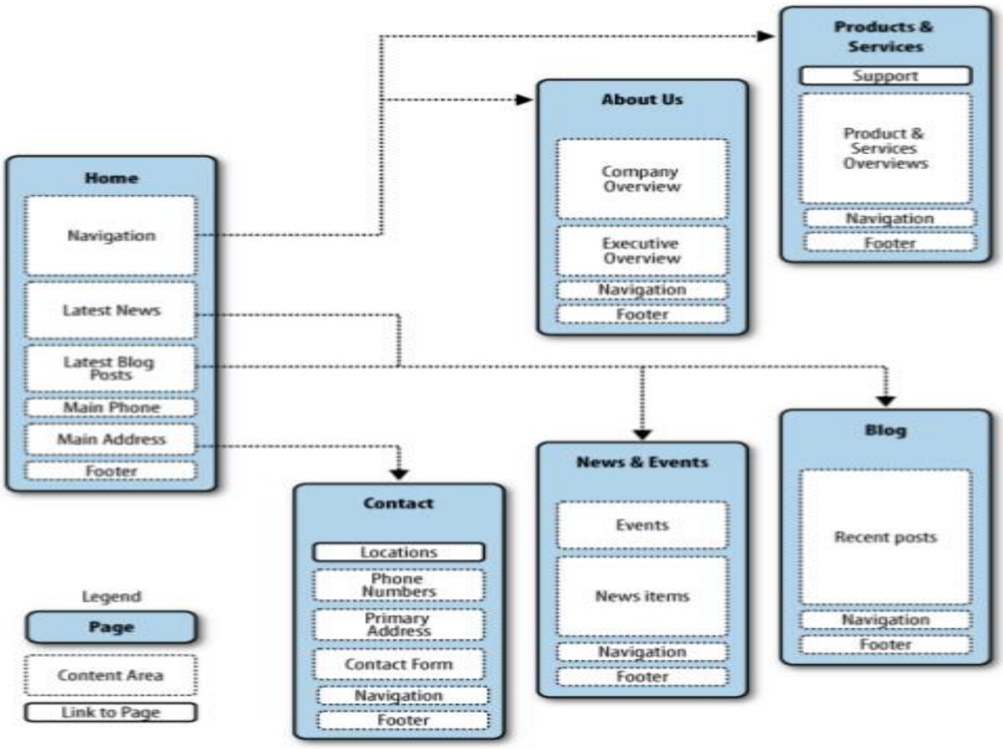


Figure 4.11. Teasing content to confirm the user's expectations of the content within

In figure4.11, we can see in a constrained screen that teasing the first few items of the page provides the user with a much more intuitive interface, immediately indicating what type of content the user can expect. We immediately saw that users were finding content more quickly, driving up our sales. It was like night and day. In Figure 4.11, you can see in a constrained screen that teasing the first few items of the page provides the user with a much more intuitive interface, immediately indicating what type of content the user can expect.
Clickstreams:
Clickstream is a term used for showing the behavior on websites, displaying the order in which users travel through a site's information architecture, usually based on data gathered from server logs. Clickstreams are usually historical, used to see the flaws in your information architecture, typically using heat-mapping or simple percentages to show where your users are going. I've always found them to be a useful tool for rear-chitecting large websites.
However, information architecture in mobile is more like software than it is the Web, meaning that createing clickstreams in the beginning, not the end. This maps the ideal path the user will take to perform common tasks. Being able to

visually lay out the path users will take gives a holistic or bird's-eye view of your mobile information architecture, just as a road map does. When all the paths are seen next to each other and take a step back, start to see shortcuts and how can get users to their goal faster or easier, as shown in figure 4.12.
Now the business analyst says, “Just create user or process flows,” such as the esoteric diagram shown in figure 4.13, which is made up of boxes and diamonds that look more like circuit board diagrams than an information architecture. A good architect's job is to create a map of user goals, not map out every technical contingency or edge case.

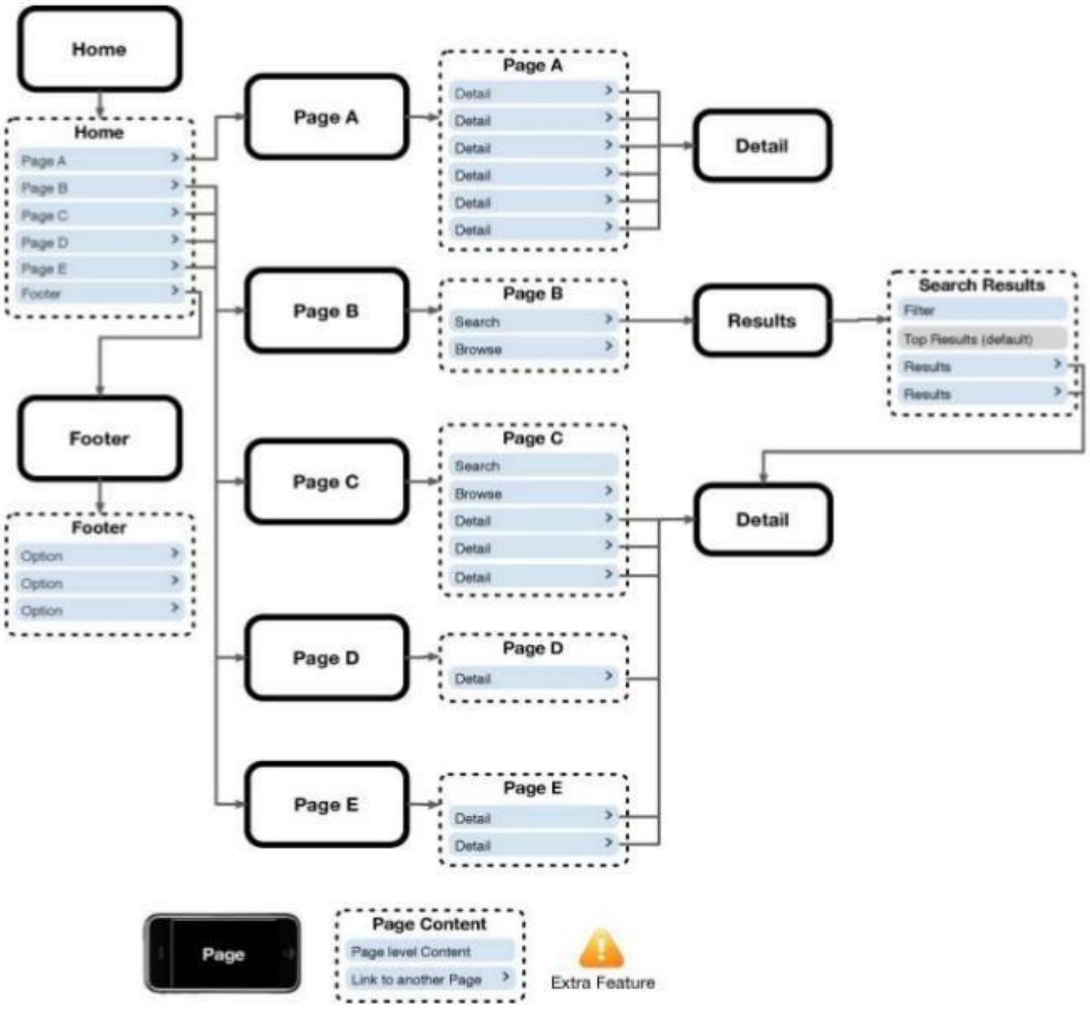


Figure 4.12. An example clickstream for an iPhone web application

Too often, process flows go down a slippery slope of adding every project requirement, bogging down the user experience with unnecessary distractions, rather than focusing on streamlining the experience. Remember, in mobile, it is to keep it as simple as possible. We need to have an unwavering focus on defining an excellent user experience first and foremost. Anything that distracts us from that goal is just a distraction.

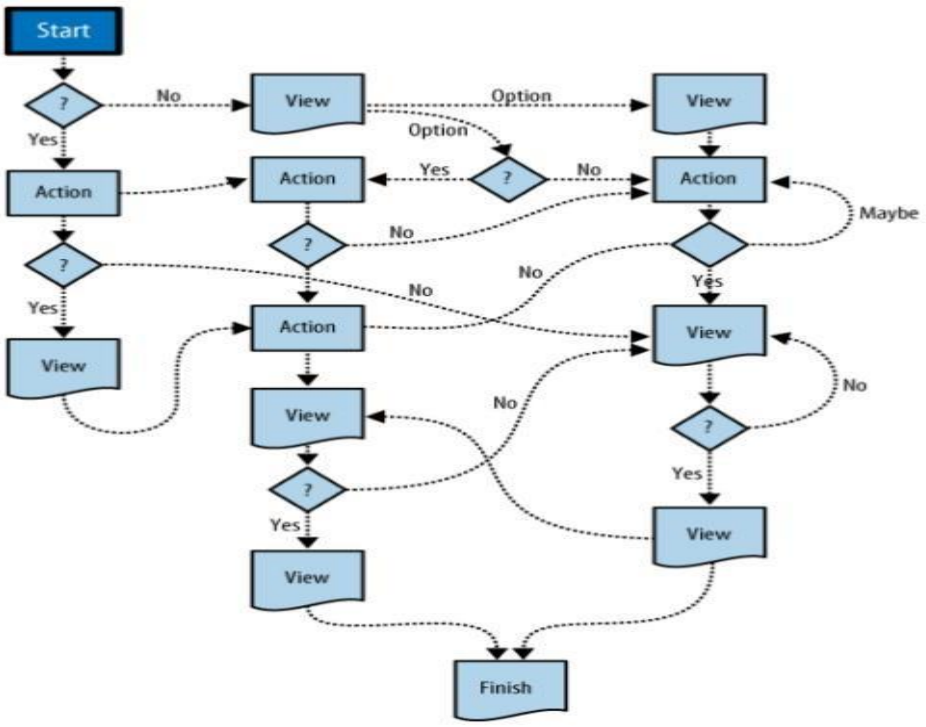


Figure 4.13. An example process flow diagram

Wireframes:

The next information architecture tool at disposal is wireframes. Wireframes are a way to lay out information on the page, also referred to as information design. Site maps show how the content is organized in our informational space; wireframes show howthe user will directly interact with it. Wireframes are like the peanut butter to the site map jelly in our information architecture sandwich. It's the stuff that sticks. Wireframes like the one in figure 4.14 a. serve to make our information space tangible and useful.

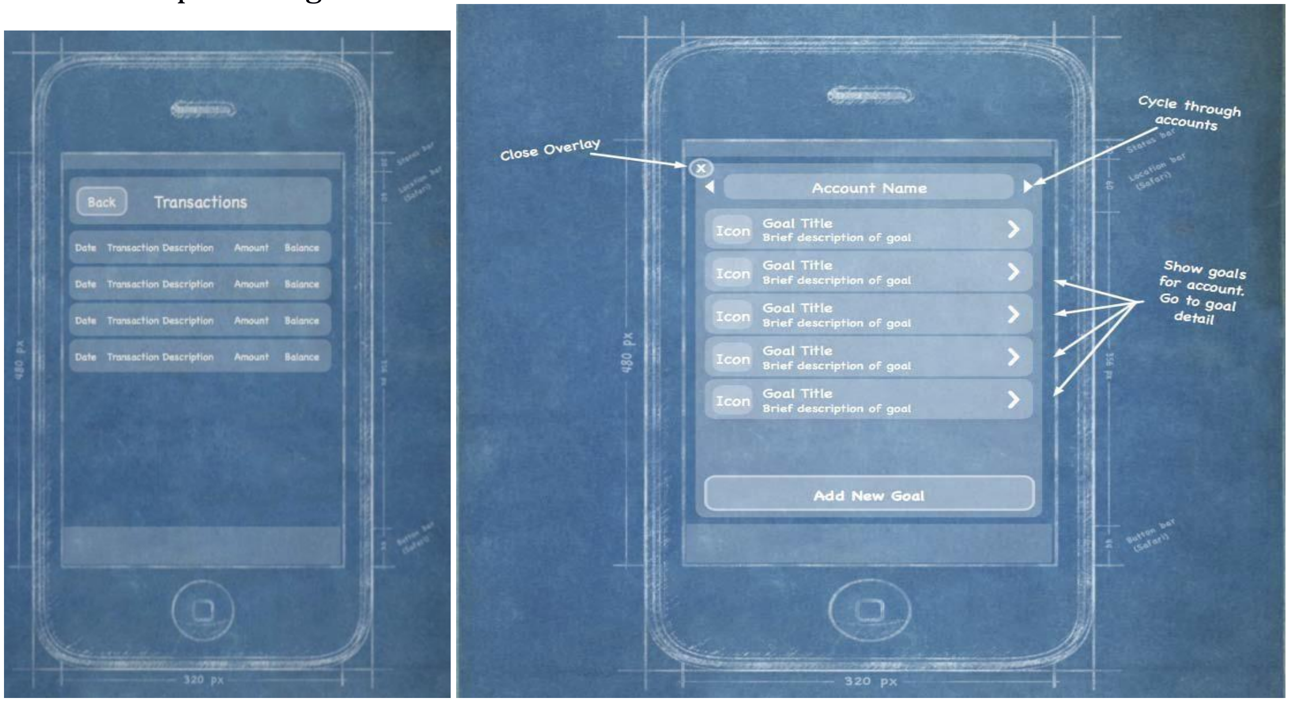

Back
Transactions
Date Transaction Description Amount Balance
Date Transaction Description Amount Balance
Date Transaction Description Amount Balance
Date Transaction Description Amount Balance
480 px
320 px
Close Overlay
Cycle through accounts
Account Name
Icon Goal Title Brief description of goal
Icon Goal Title Brief description of goal
Icon Goal Title Brief description of goal
Icon Goal Title Brief description of goal
Icon Goal Title Brief description of goal
Show goals for account. Go to goal detail
Add New Goal
480 px
320 px

Figure 4.14.

Figure 4.14. a. An example of an iPhone web application wireframe, intended to be low fidelity to prevent confusion of visual design concepts with information design concepts b. Using annotations to indicate the desired interactions of the site or application

But the purpose of wireframes is not just to provide a visual for our site map; they also serve to separate layout from visual design, defining how the user will interact with the experience. How do we lay out our navigation? What visual or interaction metaphors will we use to evoke action? These questions and many more are answered with wireframes.
Although wireframes is found to be one of the most valuable information deliverables to communicate the vision for how a site or app will work, the challenge is that a diagram on a piece of paper doesn't go a long way toward describing how the interactions will work. Most common are "in-place" interactions, or areas where the user can interact with an element without leaving the page. This can be done with Ajax or a little show/hide JavaScript. These interactions can include copious amounts of annotation, describing each content area in as much length as you can fit in the margins of the page, as shown in figure 4.14 b. In mobile, using wireframes as the key deliverable, that turns good ideas into excellent mobile products.

Prototyping:
Prototypes might sound like a scary step in the process. Some view them as redundant or too time-consuming, preferring to jump in and start coding things. But as with wireframes, that each product built out some sort of prototype has saved both time and money. The following sections discuss some ways to do some simple and fast mobile prototyping.
Paper prototypes: The most basic level we have is paper prototyping: taking our printed-out wireframes or even drawings of our interface, like the one shown in Figure 4.15, and putting them in front of people. Create a basic script of tasks (hopefully based on either context or user input) and ask users to perform them, pointing to what they would do. You act as the device, changing the screens for them. The size matters and you'll learn as much from how the user manages working with small media as you will what information is actually on it.

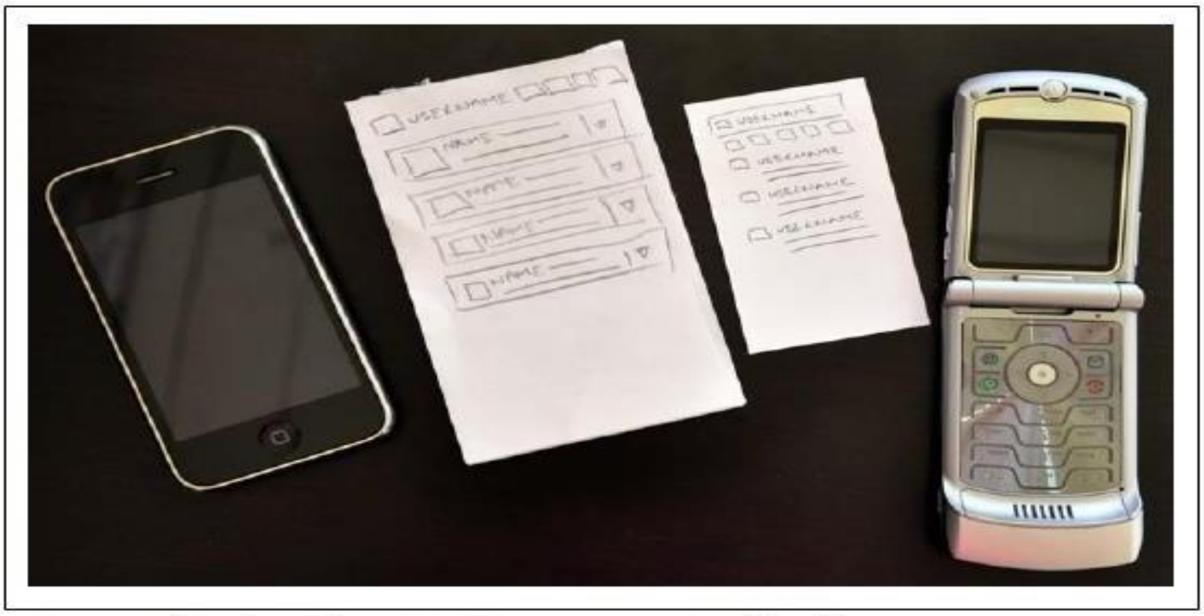
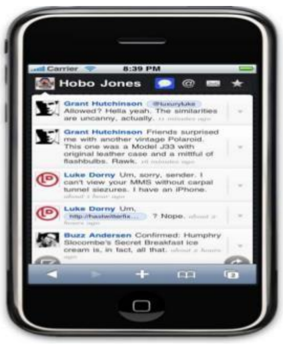


Figure 7-12. A touch interface paper prototype next to its smaller sibling

Context prototype:
The next step is creating a context prototype as in figure 4.16. Take a higher-end device that enables you to load full-screen images on it. Take your wireframes or sketches and load them onto the device, sized to fill the device screen. Leave the office. Go for a walk down to your nearest café. Or get on a bus or a train. As you are traveling about, pull out your device and start looking your interface in the various contexts you find yourself currently in. Pay particular attention to what you are thinking and your physical behaviour while you are using your interface and then write it down. If you are brave and don't have strict nondisclosure issues, ask the people around you to use it, too. I wouldn't bother with timing interactions or sessions, but try to keep an eye on a clock to determine how long the average session is.

Figure 4.15

Figure 4.16



Grant Hutchinson @luxuryluke Allowed? Hella yeah. The similarities are uncanny, actually.

Grant Hutchinson Friends surprised me with another vintage Polaroid. This one was a Model J33 with original leather case and a mitfful of flashbulbs. Rawk.

Luke Dorny Um, sorry, sender. I can't view your MMS without carpal tunnel siezures. I have an iPhone.

Luke Dorny Um, http://fastwitterfix... ? Nope.

Buzz Andersen Confirmed: Humphry Slocombe's Secret Breakfast ice cream is, in fact, all that.

Figure 4.17

HTML prototypes:
The third step is creating a lightweight, semi functional static prototype using XHTML, CSS, and JavaScript, if available. This is a prototype that you can actually load onto a device and produce the nearest experience to the final product, but with static dummy content and data as in figure 4.17. It takes a little extra time, but it is worth the effort. With a static XHTML prototype, you use all the device metaphors of navigation, you see how much content will really be displayed on screen (it is always less than you expect), and you have to deal with slow load times and network latency. In short, you will feel the same pains your user will go through. Whatever route you wish to take, building a mobile prototype takes you one very big leap forward to creating a better mobile experience.

Different Information Architecture for Different Devices:
Depending on which devices you need to support, mobile wireframes can range from the very basic to the complex. On the higher-end devices with larger screens, we might be inclined to add more interactions, buttons, and other clutter to the screen, but this would be a mistake. Just because the user might have a more advanced phone, that doesn't mean that it is a license to pack his screen with as much information.
The motivations, goals, and how users will interact with a mobile experience are the same at the low end as they are on a high-end device. For the latter, there are better tools to express the content. The greatest challenge in creating valuable experiences is knowing when to lose what is not needed. There is no choice on lower-end devices—it must be simple. When designing for both, it is best to try and to keep your information architecture as close to each other as possible without sacrificing the user experience. They say that simple design is the

hardest design, and this principle certainly is true when designing information architecture for mobile devices.

4.4 MOBILE 2.0

The mobile community started to discuss the idea of "Mobile 2.0," borrowing from many of the same principles behind Web 2.0. Each of these principles serves to transform the Web into a more agile and user-centered medium for delivering information to the masses. Mobile development, under the bottlenecks of device fragmentation and operator control, is sorely in need of a little reinvention as well.

Following is a recap of the original seven principles of Web 2.0:

The Web as a platform: For the mobile context, this means "write once, deploy everywhere," moving away from the costly native applications deployed over multiple frameworks and networks.

Harnessing collective intelligence: This isn't something the mobile community has done much of, but projects like WURFL is exactly what mobile needs more of.

Data is the next Intel inside: It can include the data we seek,the data we create, and the data about or around our physical locations.

End of the software release cycle: Long development and testing cycles heavily weigh on mobile projects, decreasing all hopes of profitability. Shorter agile cycles are needed to make mobile development work as a business. Releasing for one device, iterating, improving, and then releasing for another is a great way to ensure profitability in mobile.

Lightweight programming models: Because mobile technology is practically built on enterprise Java, the notion of using lightweight models is often viewed with some skepticism. But decreasing the programming overhead required means more innovation occurs faster.

Software above the level of a single device: This effectively means that software isn't just about computers anymore. We need to approach new software as though the user will demand it work in multiple contexts, from mobile phones to portable gaming consoles and e-book readers.

Rich user experiences: A great and rich user experience helps people spend less time with the software and more time living their lives. Mobile design is about enabling users to live their lives better.

Although the mobile industry has been through many more evolutions than just two, the concepts behind Web 2.0 are some of the most important ideas in not just mobile technology, but the Web as a whole.

Mobile 2.0: The Convergence of the Web and Mobile:

Mobile is already a medium, but the consensus is that by leveraging the power of the Web, integrating web services into the mobile medium is the future of mobile development. When the iPhone exploded onto the scene, it increased the usage of the mobile web by its users to levels never seen before. The spur of new mobile web apps created just for the iPhone doubled the number of mobile websites available in under a year. Mobile 2.0 tells us that mobile will be the primary context in which we leverage the Web in the future.

Mobile Web Applications Are the Future

Creating mobile web applications instead of mobile software applications has remained an area of significant motivation and interest. The mobile community is looking at the Web 2.0 revolution for inspiration, being able to create products and get them to market quickly and at little cost. They see the success of small

iterative development cycles and want to apply this to mobile development, something that is not that feasible in the traditional mobile ecosystem.
Developers have been keen for years to shift away from the costly mobile applications that are difficult to publish through the mobile service provider, require massive testing cycles and costly porting to multiple devices, and can easily miss the mark with users after loads of money have been dumped into them. The iPhone App Store and the other mobile device marketplaces have made it far easier to publish and sell, but developers still have to face difficult approval processes, dealing with operator and device maker terms and porting challenges.
Mobile software has two fundamental problems that mobile web applications solve.

- The first is forcing users through a single marketplace. We know from years of this model that an app sold through a marketplace can earn huge profits if promoted correctly. Being promoted correctly is the key phrase. What gets promoted and why is a nebulous process with no guarantees.
- The second problem is the ability to update your application. It is certainly possible on modern marketplaces like the App Store, but we are still years from that being the norm. Mobile web apps enable you to make sure that you never ship a broken app, or if your app breaks in the future due to a new device, to be able to fix it the same day the device hits the street. This flexibility isn't possible in the downloaded app market.

JavaScript Is the Next Frontier:
If you are going to provide mobile web applications, you have to have a mobile web browser that supports Ajax, or, as it is technically known, XMLHttpRequest. Ajax is great, but just being able to do a little show/hide or change a style after you click or touch it goes a long way toward improving the user experience. Modern mobile browsers have made much progress over the last few years, but there is still plenty of work to be done. For example, accessing the device capabilities like the phone book or filesystem with JavaScript doesn't work in a consistent way. These problems still need to be solved in order to truly reap the benefits of the Web.
Rich interactions kill battery life: JavaScript and Ajax have been ignored because using an Ajax-based web application on your phone can drain your battery at a rate of four to five times your normal power consumption. There are number of reasons for why this happens from mobile hardware guys much smarter than myself, but to summarize, the two most prevalent are:
• JavaScript consumes more processor power and therefore more battery life.
• Ajax apps fetch more data from the network, meaning more use of the radio and more battery life.
Apple and the open source WebKit browser have made huge strides by releasing a JavaScript engine that is incredibly efficient on mobile devices, though the other big mobile browser technologies aren't far behind. This problem is going away quickly as the mobile browsers get better, batteries improve efficiency, and devices get more powerful.
The Mobile User Experience Is Awful:
Device API's usually force to use their models of user experience, meaning that have to work in the constraints of the API. This is good for creating consistent experiences for that platform, but these experiences don't translate to others. For example, would you take an iPhone app design and put it on an Android device? The user experience for these devices is similar but still remains different. Modern mobile web browsers, as they come closer to their desktop counterparts,

remove this distinction, giving us the same canvas on mobile devices that we have for the desktop. It means we can have a consistent user experience across multiple mediums.

Mobile Widgets Are the Next Big Thing:

At many Mobile 2.0 events, there are a lot of buzz about mobile widgets, though no one can tell how mobile widgets would define a mobile widget, or how they are different from mobile web apps. The consensus seems to be that the solution for the challenges with the mobile web is to create a series of "small webs" targeted at a specific user or task. The concept of small network-enabled applications is very promising, but the mobile industry tends to take promising ideas like this, inflate expectations to unsustainable levels, then abandon them at the first sign of trouble or sacrifice them for the next big thing, whichever happens first.

Carrier Is the New "C" Word:

It is clear that one of the key drivers of Mobile 2.0 and the focus on the mobile web is to find a way to build a business that doesn't rely on carrier control.

Mobile Needs to Check Its Ego:

On the mobile side, there are some incredibly intelligent people who have been innovating amazing products under insane constraints for years. On the web side, there are creative amateurs who have helped build a community and ecosystem out of passion and an openness to share information.

The web guys want to get into the game and move the medium forward, partly out of desire open up a new market for themselves, but mostly out of passion for all things interactive. But, to the mobile community, they are seen as a threat to expertise. On the other hand, to the web community, the mobile guys come off as overly protective, territorial, selfish, and often snobbish or egotistical, effectively saying, "Go away." They have to deal with really hard problems that would make a web professional give up to go serve coffee.

Unless the mobile community comes together with the web community by sharing information, experience, and guidance, one day they will find that their experience has become obsolete. In return, the web guys will share their enthusiasm, willingness to learn, and passion that many in mobile development have forgot. It's that one principle of Web 2.0 that the mobile community has left out: harnessing collective intelligence. The Web and the mobile community are reaching a point where the two worlds can no longer afford not to be working together, sharing what they know and harnessing the collective intelligence of both media.

We Are Creators, Not Consumers:

The final principle of Mobile 2.0 is recognizing that we are in a new age of consumerism. Yesterday's consumer does not look anything like today's consumer. The people of today's market don't view themselves as consumers, but rather as creators. But before we get into that, let's back up for a minute. The web is about content. Sure, there are programming languages, APIs, and other technical underpinnings, but what do you do when you open a web browser? You read.

Our primary task online is to read, to gain information. During the early days of the Web, it took tools and know-how in order to publish to the Web. But early in the Web 2.0 evolution, we saw a rise in tools that allowed us to publish to the Web easily, giving individuals a voice online, with a massive audience.

This democratization of the Web took many forms that some call "social media," like blogging, social networks, media sharing, microblogging, and lifestreams.

Although social media may have many facets, they all share the same goal: to empower normal, everyday people to become creators and publishers of content. It started with the written word, then music, then photos, and more recently video was added. Entire markets have been created to provide today's consumer with gadgets, software, and web services to record and publish content so that we can share it with our friends and loved ones. At the center of this revolution in publishing is the mobile device. As networked portable devices become more powerful, allowing us to capture, record, and share content in the moment, we are able to add a new kind of context to content—the likes of which
we haven't seen since satellite television. Now you can share any moment with any group of people in real time. Think about how powerful a concept that is! It could change entire cultures. Tony Fish, coauthor of Mobile Web 2.0 (futuretext), says: When everyone has the tools to create content, in addition to zero-cost publishing, we do not consume content, we create it. In the early days of the Web, I marveled at how a networked population might change our society forever. Now I realize that the change occurs wherever the device is, the context it is within. The early "Web 1.0" days clearly changed how business is done, because businesses are the primary consumer of desktop computers. It probably is no coincidence that Web 2.0 occurred around the same time that laptop computers became affordable for the average person, making the Web a more personal medium. With Mobile 2.0, the personal relevance of the content matches how personal the device is and how personally it applies to our everyday situations or our context. I see now that this is the time and medium that delivers on that initial promise of the Web: to change society forever.

4.5 MOBILE DESIGN

Interpreting Design:

Mobile design isn't that different. Precise designs might look better, but they can be brutal to implement. More flexible designs might not be much to look at, but they work for the most users, or the lowest common denominator. But more than that, our back-grounds and our training can actually get in the way of creating the best design for the medium. We like to apply the same rules to whatever the problem in front of us might be. In mobile design, you interpret what you know about good design and translate it to this new medium that is both technologically precise and at times incredibly unforgiving, and you provide the design with the flexibility to present information the way you envision on a number of different devices.

The Mobile Design Tent-Pole:

To have a successful mobile design, we have to adapt to today's changing audiences and niches. Find that emotional connection, that fundamental need that serves many audiences, many cultures, and many niches and design experiences. Too often, designers simply echo the visual trends of the day, mimicking the inspiration of others. But with mobile design, once you find that essential thing, that chewy nougat we call "context" that lives at the center of your product, then you will find ample inspiration of your own to start creating designs that translate not only across multiple devices, but across multiple media.

Sure, there are countless examples of poorly designed mobile products that are considered a success. You only need to look as far as the nearest mobile app store to find them. This is because of the sight unseen nature of mobile

commerce. Traditionally, you couldn't demo—or in some cases even see—screenshots of a game or mobile application before you bought it. You simply had to purchase it and find out if it was any good.
Apple's App Store quickly changed that. We can clearly see that the best-selling games and applications for the iPhone are the ones with the best designs (Figure 4.18).Users look at multiple screenshots, read the user reviews, and judge the product based on the quality of its icon and of the screenshots before they buy. The Apple App Store is proving everyday that mobile design doesn't have to start with tent-pole lowest-common-denominator products—it can instead start with providing the best possible experience and tailoring that experience to the market that wants it most.

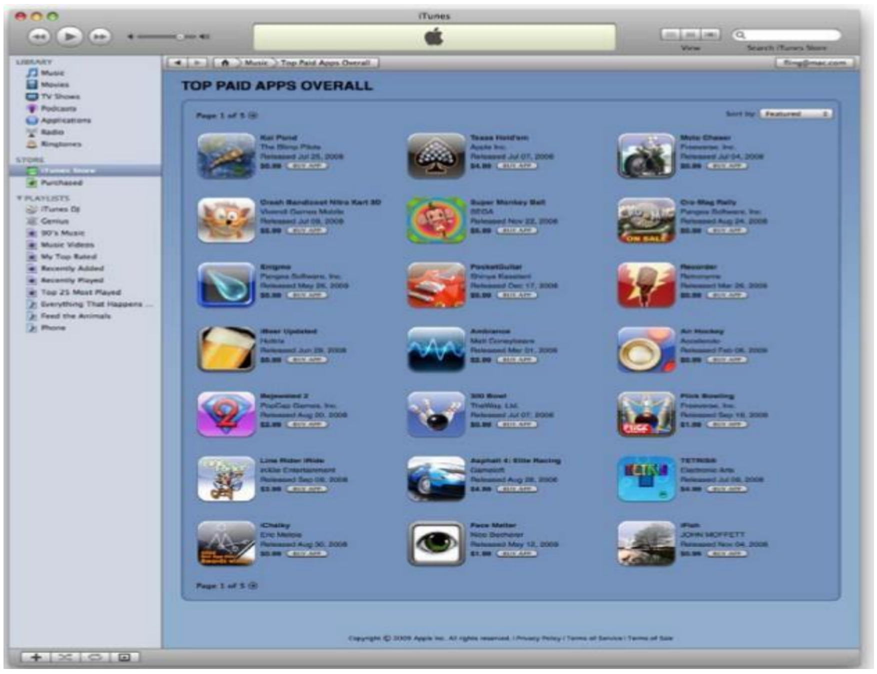


Figure 4.18. The app icon design greatly influences the user's expectation of quality

Designing for the Best Possible Experience:
When the first iPhone came out, there was a lot of trouble from web and mobile peers for publicly saying, "The iPhone is the only mobile device that matters right now." They would argue, "What about ABC or XYZ platforms?". The response was that those are important, but the iPhone provides the best possible experience and that is where consumers will go. Since those days, the iPhone shatter just about every record in mobile devices, becoming one of the best-selling phones ever and one of the most used mobile browsers in the world—two-thirds of mobile browsing in the U.S. comes from an iPhone or an iPod touch, not to mention that more than a billion mobile applications have been sold for these devices in under a year.
Although it may defy the business instincts to focus the product on just one device, in mobile development, the risks and costs of creating that tent-pole product are just too high. This is so easily seen through bad or just plain uninspired mobile design. Asking creative people to create uninspiring work is a fast track to mediocrity.
Here is a design solution: design for the best possible experience. Actually, don't just design for it: focus on creating the best possible experience with unwavering

passion and commitment. Iterate, tweak, and fine-tune until you get it right. Anything less is simply unacceptable. Do not get hindered by the constraints of the technology. Phrases like "lowest common denominator" cannot be part of the designer's vocabulary. Your design—no, your work of art—should serve as the shining example of what the experience should be, not what it can be. Trying to create a mobile design in the context of the device constraints isn't where you start; it is where you should end.
The greatest mistakes we in the mobile community make is being unwilling to or feeling incapable of thinking forward. The tendency to frame solutions in the past (past devices, past standards) applies only to those low-quality, something-for-everyone-but-getting-nothing tent-pole products. Great designs are not unlike great leaps forward in innovation. They come from shedding the baggage regarding how things are done and focus on giving people what they want or what they need.

4.5.1 ELEMENTS OF MOBILE DESIGN

Good mobile design requires three abilities: the first is a natural gift for being able to see visually how something should look that produces a desired emotion with the target audience. The second is the ability to manifest that vision into something for others to see, use, or participate in. The third is knowing how to utilize the medium to achieve your design goals.

Context:
As the designer, it is the job to make sure that the user can figure out how to address context using your app. Make sure to answer the following questions:
• Who are the users? What do you know about them? What type of behavior can you assume or predict about the users?
• What is happening? What are the circumstances in which the users will best absorb the content you intend to present?
• When will they interact? Are they at home and have large amounts of time? Are they at work where they have short periods of time? Will they have idle periods of time while waiting for a train, for example?
• Where are the users? Are they in a public space or a private space? Are they inside or outside? Is it day or is it night?
• Why will they use your app? What value will they gain from your content or services in their present situation?
• How are they using their mobile device? Is it held in their hand or in their pocket? How are they holding it? Open or closed? Portrait or landscape?
The answers to these questions will greatly affect the course of your design. Treat these questions as a checklist to your design from start to finish. They can provide not only great inspiration for design challenges, but justification for your design decisions later.

Message:
Another design element is your message, or what you are trying to say about your site or application visually. One might also call it the "branding," although I see branding and messaging as two different things. The message is the overall mental impression create explicitly through visual design. If we take a step back, and look at a design from a distance, what is our impression? Or conversely, look at a design for 30 seconds, and then put it down. What words would we use to describe the experience?

Branding shouldn't be confused with messaging. Branding is the impression your company name and logo gives—essentially, your reputation. Branding serves to reinforce the message with authority, not deliver it. In mobile, the opportunities for branding are limited, but the need for messaging is great. With such limited real estate, the users don't care about the brand, but they will care about the messaging, asking themselves questions like, "What can this do for me?" or "Why is this important to me?"

The approach to the design will define that message and create expectations. A sparse, minimalist design with lots of whitespace will tell the user to expect a focus on content. A "heavy" design with use of dark colors and lots of graphics will tell the user to expect something more immersive. For example, hold the book away from you and look at each of the designs in Figure 4.19; try not to focus too heavily on the content. What do each of these designs "say" to you?

Which of the following designs provide a message? What do they say to you?

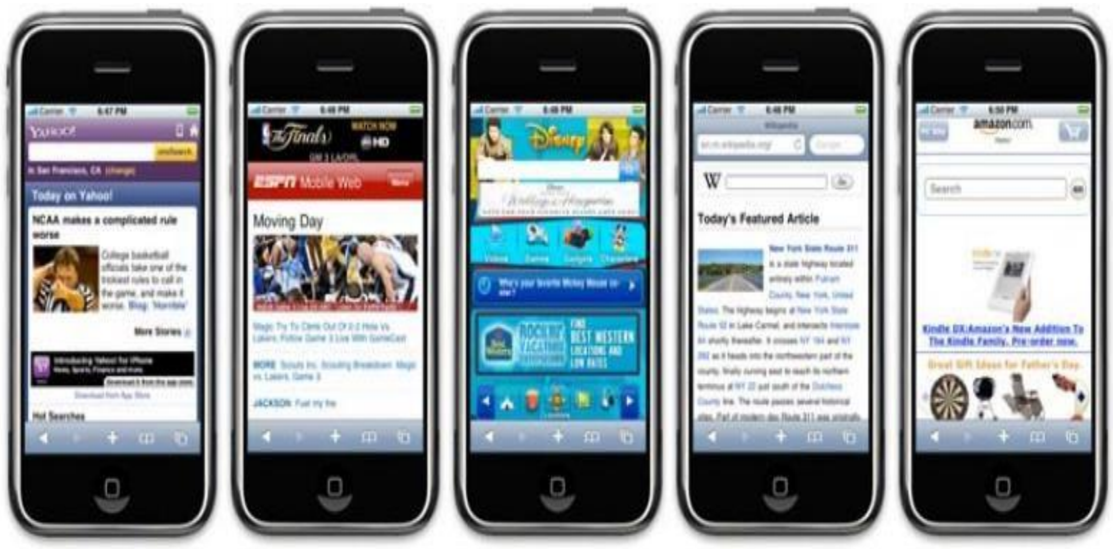


Figure 4.19. What is the message for each of these designs?

What is the message for each of these designs?

Yahoo!: Yahoo! sort of delivers a message. This app provides a clean interface, putting a focus on search and location, using color to separate it from the news content. But I'm not exactly sure what it is saying. Words you might use to describe the message are crisp, clean, and sharp.

ESPN: The ESPN site clearly is missing a message. It is heavily text-based, trying to put a lot of content above the fold, but doesn't exactly deliver a message of any kind. If you took out the ESPN logo, you likely would have indifferent expectations of this site; it could be about anything, as the design doesn't help set expectations for the user in any way. Words you might use to describe the message: bold, cluttered, and content-heavy.

Disney: Disney creates a message with its design. It gives you a lot to look at—probably too much—but it clearly tries to say that the company is about characters for a younger audience. Words you might use to describe the message: bold, busy, and disorienting.

Wikipedia: The Wikipedia design clearly establishes a message. With a prominent search and text-heavy layout featuring an article, you know what you are getting with this design. Words you might use to describe the message: clean, minimal, and text-heavy.

Amazon: Amazon sort of creates a message. Although there are some wasted opportunities above the fold with the odd ad placement, you can see that it is mostly about products (which is improved even more if you scroll down). Words you might use to describe the message: minimal but messy, product-heavy, and disorienting.

Look and Feel:

The concept of "look and feel" is an odd one, being subjective and hard to define. Typically, look and feel is used to describe appearance, as in "I want a clean look and feel" or "I want a usable look and feel." The problem is: as a mobile designer, what does it mean? And how is that different than messaging?

Look and feel in a literal sense, as something real and tactile that the users can "look" at, then "feel"—something they can touch or interact with. Look and feel is used to evoke action—how the user will use an interface. Messaging is holistic, as the expectation the users will have about how you will address their context. It is easy to confuse the two, because "feel" can be interpreted to mean our emotional reaction to design and the role of messaging.

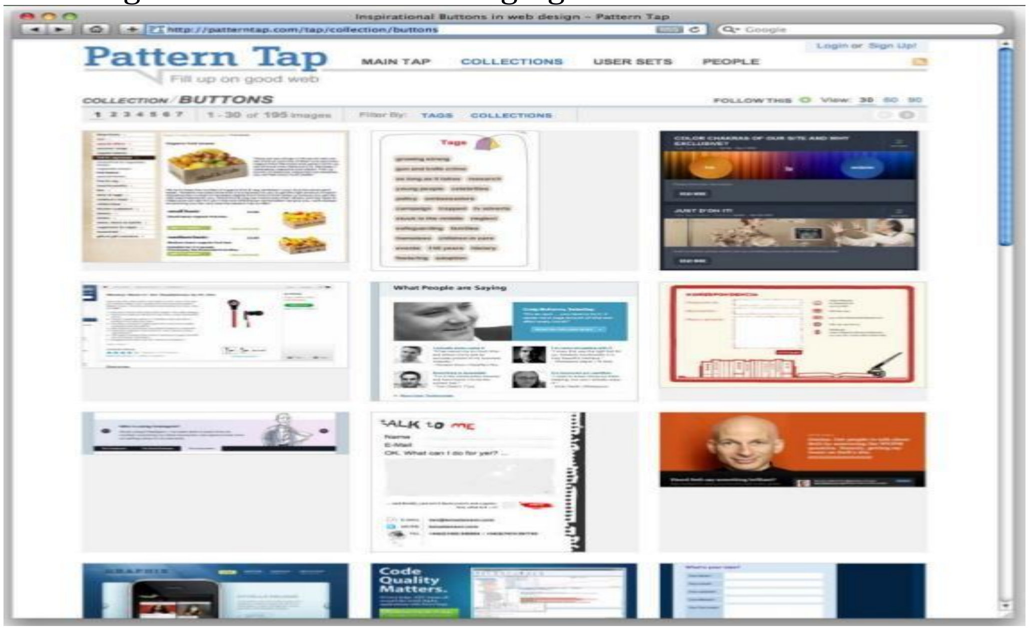


Figure 4.20: Pattern Tap shows a number of user interface patterns that help to establish look and feel

On large mobile projects or in companies with multiple designers, a style guide or pattern library is crucial, maintaining consistency in the look and feel and reducing the need for each design decision to be justified. For example, in Figure 4.20 you can see the site Pattern Tap, which is a visual collection of many user interface patterns meant for websites and web applications, but there is no reason why it can't serve as inspiration for your mobile projects as well. Although a lot of elements go into making Apple's App Store successful, the most important design element is how it looks and feels. Apple includes a robust user interface tool that enables developers to use prebuilt components, supported with detailed Human Interface Guidelines (or HIG) of how to use them, similar to a pattern library. This means that a developer can just sit down and create an iPhone application that looks like it came from Apple in a matter of minutes. During the App Store submission process, Apple then ensures that the developer uses these tools correctly according to the HIG. The look and feel can either be consistent with the stock user interface elements that Apple provides; they can be customized, often retaining the “spirit” of Apple’s original design; or an entirely new look and feel can be defined—this approach is often used for immersive experiences. The stock user experience that Apple provides is a great example of how look and feel works to supporting messaging.

For the end user, the design sends a clear message: by using the same visual interface metaphors that Apple uses throughout the iPhone, I can expect the action, or how this control will behave, but I can also expect the same level of quality. This invokes the message of trust and quality in the application and in the platform as a whole. Apple isn’t the first to use this shared look and feel model in mobile—in fact, it is incredibly common with most smartphone platforms—but they are surely making it incredibly successful, with a massive catalog of apps and the sales to support it.
The mobile designers must be creative and remember the context. Like in the early days of the Web, people tend to be skeptical about mobile experiences. The modal context of the user—in this case, what device he is using—should be considered during the design, as it will help to establish the user’s expectations of the experience.

Layout:
Layout is an important design element, because it is how the user will visually process the page, but the structural and visual components of layout often get merged together, creating confusion and making your design more difficult to produce. The first time layout should rear its head is during information architecture. In fact, about 90 percent of layout decisions were during the information architecture period. We have to ask ourself questions like: where should the navigation go on the page or screen? What kind of navigation type should I use? Should I use tabs or a list? What about a sidebar for larger screens? All of these should be answered when defining the information architecture and before you begin to design. Why define the layout before the mobile design? Design is just too subjective of an issue. If you are creating a design for anyone but yourself, chances are good that there will be multiple loosely-based-on-experience opinions that will be offered and debated. There is no right answer—only opinions and gut instincts. Plus, in corporate environments you have internal politics you have to consider, where the design opinions of the CEO or Chief Marketing Officer (CMO) might influence a design direction more than, say, the Creative Director or Design Director.
By defining design elements like layout prior to actually applying the look and feel, we can separate the discussion. As a self-taught designer, we started out in this business making designs for my own projects. I could just put pen to paper and tweak it to my heart’s content. If I wanted to radically change the layout, I could. When I started my mobile design career with my first mobile company more than a decade ago, I realized that this approach didn’t work. The majority of comments that reviewers would make were about the layout. They focused on the headers, the navigation, the footer, or how content blocks are laid out, and so on. But their feedback got muddied with the “look and feel, the colors, and other design elements.” Reviewers do make remarks like “I like the navigation list, but can you make it look more raised?” Most designers don’t hear that; they hear “The navigation isn’t right, do it again.” But, with this kind of feedback, there are

two important pieces of information about different types of design. First, there is confirmation that the navigation and layout are correct. Second, there is a question about the "look and feel." Because designers hear "Do it again," they typically redo the layout, even though it was actually fine.
Creating mobile designs in an environment with multiple reviewers is all about getting the right feedback at the right time. Your job is to create a manifestation of a shared vision. Layout is one of the elements you can present early on and discuss in-dependently. People confuse the quality and fidelity of your deliverables as design. By keeping it basic, you don't risk having reviewers confuse professionalism with design.
The irony is that as I become more adept at defining layouts, I make them of increasingly lower fidelity. For example, when I show my mobile design layouts as wireframes during the information architecture phase, I intentionally present them on blueprint paper, using handwriting fonts for my annotations (Figure 4.21). It also helps to say that this is not a design, it is a layout, so please give me feedback on the layout.

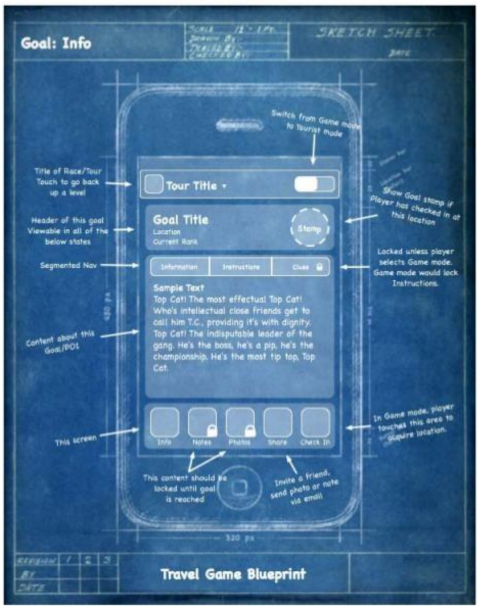

Goal: Info
SKETCH SHEET
Switch from Game mode to Tourist mode
Title of Race/Tour Touch to go back up a level
Tour Title
Show Goal stamp if Player has checked in at this location
Header of this goal Viewable in all of the below states
Goal Title
Location
Current Rank
Stamp
Segmented Nav
Information
Instructions
Clues
Locked unless player selects Game mode. Game mode would lock Instructions.
Sample Text
Top Cat! The most effectual Top Cat! Who's intellectual close friends get to call him T.C., providing it's with dignity. Top Cat! The indisputable leader of the gang. He's the boss, he's a pip, he's the championship. He's the most tip top, Top Cat.
Content about this Goal/POI
480 px
This screen
Info
Notes
Photos
Share
Check In
In Game mode, player touches this area to acquire location.
This content should be locked until goal is reached
Invite a friend, send photo or note via email
320 px
Travel Game Blueprint

Figure 4.21. Using a low-fidelity wireframe to define the layout design element before visual design begins

Different layouts for different devices: The second part of layout design is how to visually represent content. In mobile design, the primary content element you deal with the is navigation. Whether you are designing a site or app, you need to provide users with methods of performing tasks, navigating to other pages, or reading and interacting with content. This can vary, depending on the devices you support. There are two distinct types of navigation layouts for mobile devices: touch and scroll. With touch, you literally point to where you want to go; therefore, navigation can be anywhere on the screen. But we tend to see most of the primary actions or navigation areas living at the bottom of the screen and secondary actions living at the top of the screen, with the area in between serving as the content area, like what is shown in Figure 4.22.
This is the opposite of the scroll navigation type, where the device's D-pad is used to go left, right, up, or down. When designing for this type of device, the primary and often the secondary actions should live at the top of the screen. This is so the user doesn't have to press down dozens of times to get to the important stuff. In

Figure 4.23, you can actually see by the bold outline that the first item selected on the screen is the link around the logo.

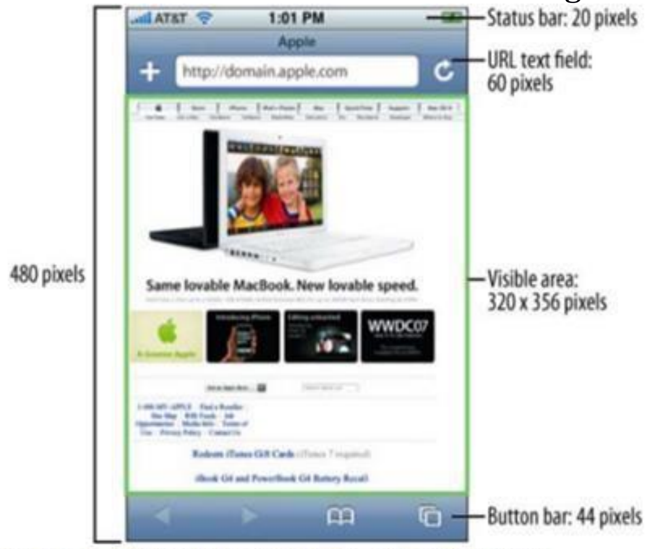


Figure 4.22. iPhone HIG, showing the layout dimensions of Safari on the iPhone.

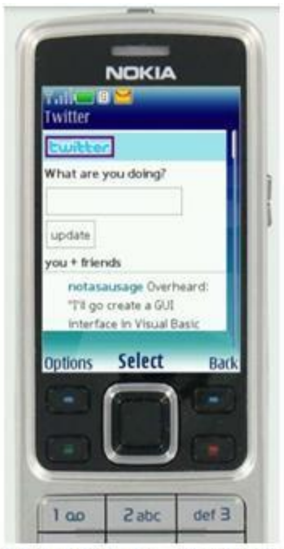
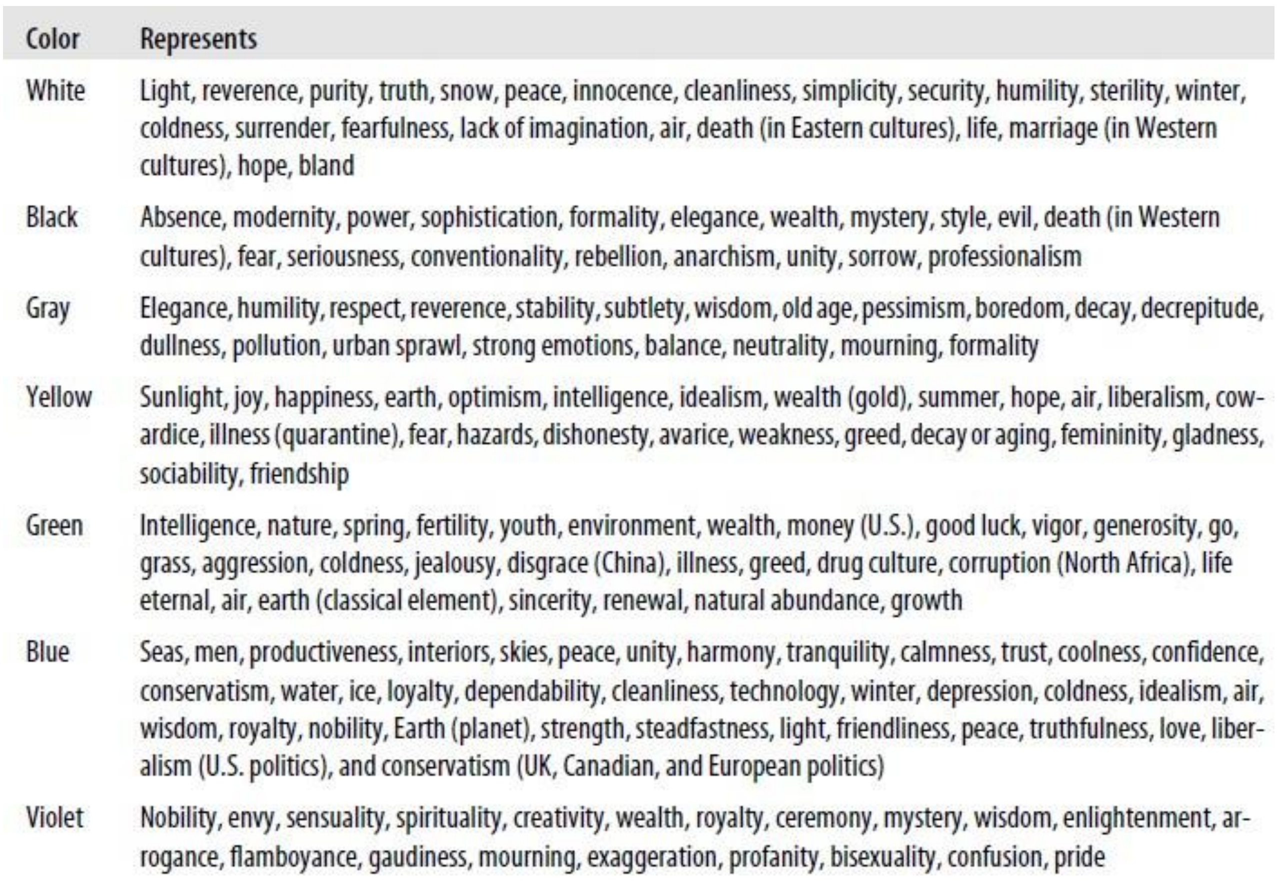


Figure 4.23. Example layout of a scroll-based application, where the user had to press the D-pad past each link to scroll the page

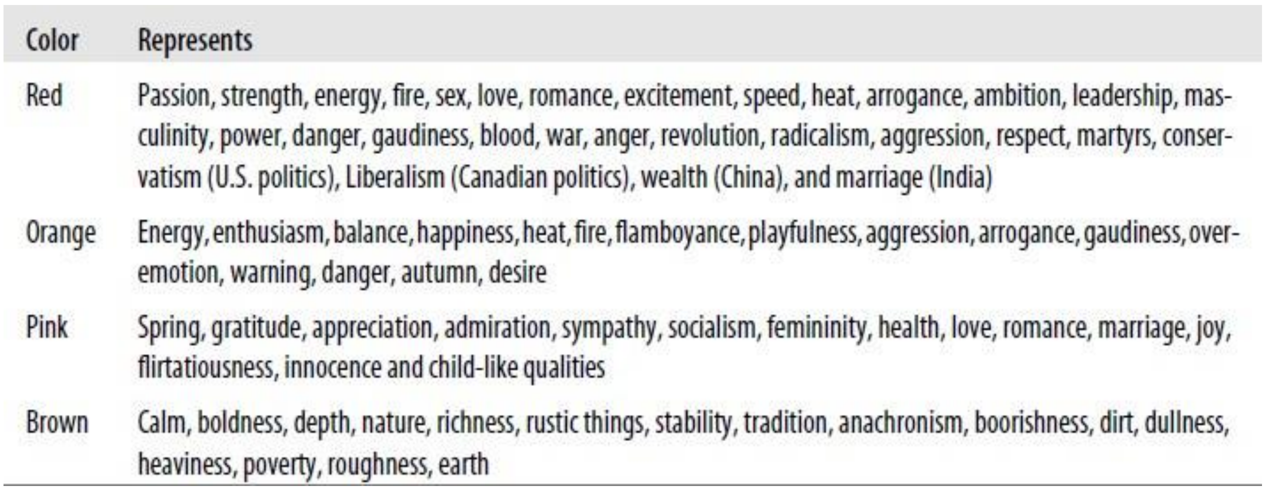
When dealing with scroll navigation, you also have to make the choice of whether to display navigation horizontally or vertically. Visually, horizontally makes a bit more sense, but when you consider that it forces the user to awkwardly move left and right, it can quickly become a bit cumbersome for the user to deal with. There is no right or wrong way to do it, but my advice is just to try and keep it as simple as possible. Fixed versus fluid Another layout consideration is how your design will scale as the device orientation changes, for example if the device is rotated from portrait mode to landscape and vice versa. This is typically described as either being fixed (a set number of pixels wide), or fluid (having the ability to scale to the full width of the screen regardless of the device orientation).
Orientation switching has become commonplace in mobile devices, and your design should always provide the user with a means to scale the interface to take full advantage of screen real estate.

Color
The fifth design element, color, is hard to talk about in a black-and-white book. Maybe it is fitting, because it wasn't that long ago that mobile screens were available only in black and white (well, technically, it was black on a green screen). These days, we have nearly the entire spectrum of colors to choose from for mobile designs. The most common obstacle you encounter when dealing with color is mobile screens, which come in a number of different color or bit depths, meaning the number of bits (binary digits) used to represent the color of a single pixel in a bitmapped image. When complex designs are displayed on different mobile devices, the limited color depth on one device can cause banding, or unwanted posterization in the image. Different devices have different color depths.

Color characteristics



Color	Represents
White	Light, reverence, purity, truth, snow, peace, innocence, cleanliness, simplicity, security, humility, sterility, winter, coldness, surrender, fearfulness, lack of imagination, air, death (in Eastern cultures), life, marriage (in Western cultures), hope, bland
Black	Absence, modernity, power, sophistication, formality, elegance, wealth, mystery, style, evil, death (in Western cultures), fear, seriousness, conventionality, rebellion, anarchism, unity, sorrow, professionalism
Gray	Elegance, humility, respect, reverence, stability, subtlety, wisdom, old age, pessimism, boredom, decay, decrepitude, dullness, pollution, urban sprawl, strong emotions, balance, neutrality, mourning, formality
Yellow	Sunlight, joy, happiness, earth, optimism, intelligence, idealism, wealth (gold), summer, hope, air, liberalism, cowardice, illness (quarantine), fear, hazards, dishonesty, avarice, weakness, greed, decay or aging, femininity, gladness, sociability, friendship
Green	Intelligence, nature, spring, fertility, youth, environment, wealth, money (U.S.), good luck, vigor, generosity, go, grass, aggression, coldness, jealousy, disgrace (China), illness, greed, drug culture, corruption (North Africa), life eternal, air, earth (classical element), sincerity, renewal, natural abundance, growth
Blue	Seas, men, productiveness, interiors, skies, peace, unity, harmony, tranquility, calmness, trust, coolness, confidence, conservatism, water, ice, loyalty, dependability, cleanliness, technology, winter, depression, coldness, idealism, air, wisdom, royalty, nobility, Earth (planet), strength, steadfastness, light, friendliness, peace, truthfulness, love, liberalism (U.S. politics), and conservatism (UK, Canadian, and European politics)
Violet	Nobility, envy, sensuality, spirituality, creativity, wealth, royalty, ceremony, mystery, wisdom, enlightenment, arrogance, flamboyance, gaudiness, mourning, exaggeration, profanity, bisexuality, confusion, pride



Color	Represents
Red	Passion, strength, energy, fire, sex, love, romance, excitement, speed, heat, arrogance, ambition, leadership, masculinity, power, danger, gaudiness, blood, war, anger, revolution, radicalism, aggression, respect, martyrs, conservatism (U.S. politics), Liberalism (Canadian politics), wealth (China), and marriage (India)
Orange	Energy, enthusiasm, balance, happiness, heat, fire, flamboyance, playfulness, aggression, arrogance, gaudiness, overemotion, warning, danger, autumn, desire
Pink	Spring, gratitude, appreciation, admiration, sympathy, socialism, femininity, health, love, romance, marriage, joy, flirtatiousness, innocence and child-like qualities
Brown	Calm, boldness, depth, nature, richness, rustic things, stability, tradition, anachronism, boorishness, dirt, dullness, heaviness, poverty, roughness, earth

The psychology of color: People respond to different colors differently. It is fairly well known that different colors produce different emotions in people, but surprisingly few talk about it outside of art school. Thinking about the emotions that colors evoke in people is an important aspect of mobile design, which is such a personal medium that tends to be used in personal ways. Using the right colors can be useful for delivering the right message and setting expectations.

Color palettes: Defining color palettes can be useful for maintaining a consistent use of color in your mobile design. Color palettes typically consist of a predefined number of colors to use throughout the design. Selecting what colors to use varies from designer to designer, each having different techniques and strategies for deciding on the colors. I’ve found that I use three basic ways to define a color palette:

i) **Sequential:** In this case, there are primary, secondary, and tertiary colors. Often the primary color is reserved as the “brand” color or the color that most closely resembles the brand’s meaning. The secondary and tertiary colors are often complementary colors that I select using a color wheel.

ii) **Adaptive:** An adaptive palette is one in which you leverage the most common colors present in a supporting graphic or image. When creating a design that is meant to look native on the device, I use an adaptive palette to make sure that my colors are consistent with the target mobile platform.

iii) **Inspired:** This is a design that is created from the great pieces of design you might see online, or offline, in which a picture of the design might inspire you. This could be anything from an old poster in an alley, a business card, or some packaging. When I sit down with a new design, I thumb through some of materials to create an inspired palette. Like with the adaptive palette, you actually extract the colors from the source image, though you should never ever use the source material in a design.

Typography

The sixth element of mobile design is typography, which in the past would bring to mind the famous statement by Henry Ford:

Any customer can have a car painted any color that he wants so long as it is black. Traditionally in mobile design, you had only one typeface that you could use (Figure 8-12), and that was the device font. The only control over the presentation was the size.

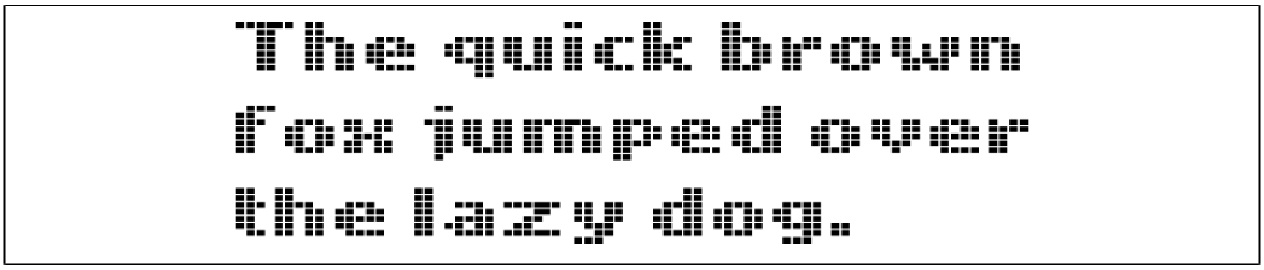


Figure 8-12. What most mobile designers think of when it comes to mobile typography

As devices improved, so did their fonts. Higher-resolution screens allowed for a more robust catalog of fonts than just the device font. First, let’s understand how mobile screens work.

Subpixels and pixel density: There seem to be two basic approaches to how type is rendered on mobile screens: using subpixel-based screens or having a greater pixel density or pixels per inch (PPI). A subpixel is the division of each pixel into a red, green, and blue (or RGB) unit at a microscopic level, enabling a greater level of antialiasing for each font character or glyph. The addition of these RGB subpixels enables the eye to see greater variations of gray, creating sharper antialiasing and crisp text.

In Figure 4.24, you can see three examples of text rendering. The first line shows a simple black and white example, the second shows text with grayscale antialiasing, and the third line shows how text on a subpixel display would render.

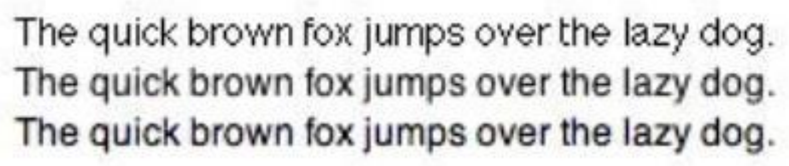
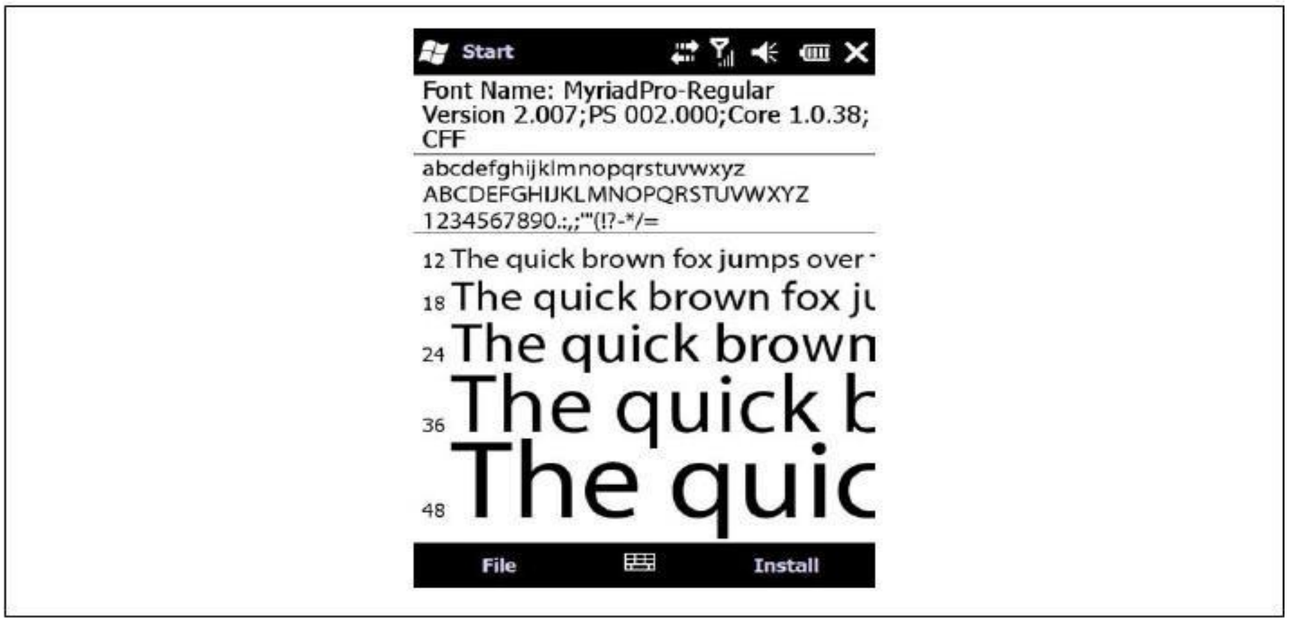

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

Figure 4.24. Different ways text can render on mobile screens


Font Name: MyriadPro-Regular
Version 2.007;PS 002.000;Core 1.0.38;
CFF

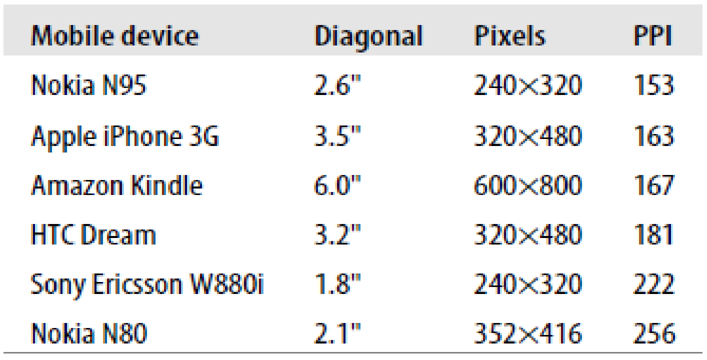
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890.:,;'"(!?-*/=

12 The quick brown fox jumps over
18 The quick brown fox ju
24 The quick brown
36 The quick b
48 The quic

Figure 8-14. Microsoft ClearType using subpixels to display sharp text

The Microsoft Windows Mobile platform uses the subpixel technique with its Clear-Type technology.

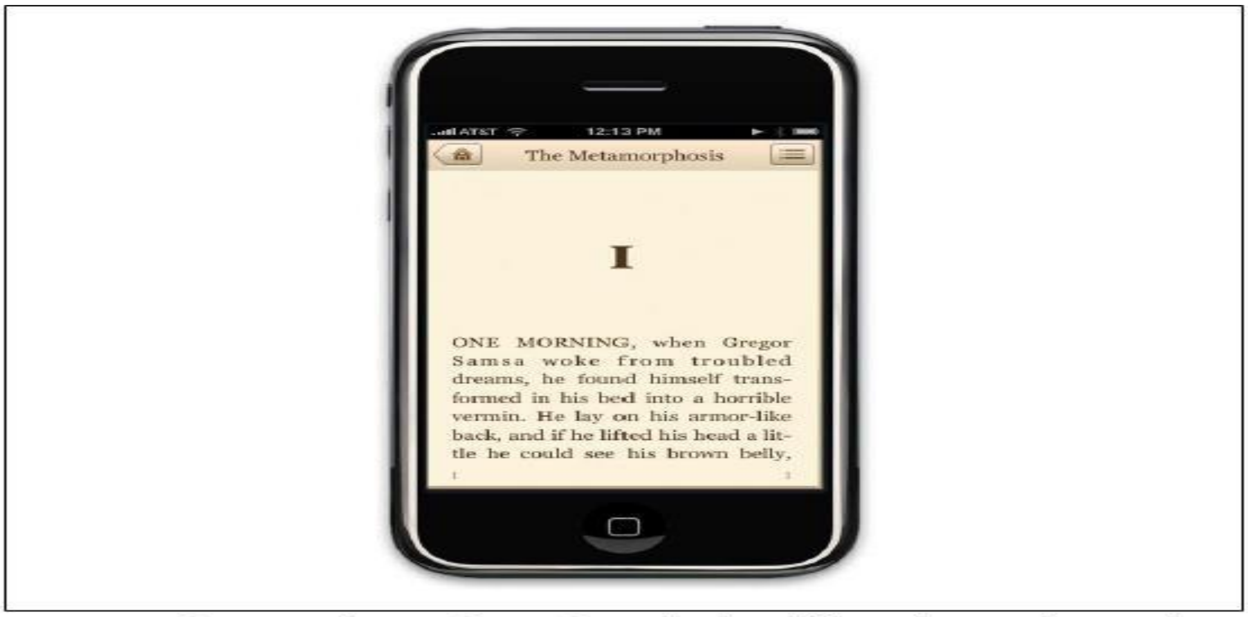
Table 8-3. Dimensions and PPI for some mobile devices



Mobile device	Diagonal	Pixels	PPI
Nokia N95	2.6"	240×320	153
Apple iPhone 3G	3.5"	320×480	163
Amazon Kindle	6.0"	600×800	167
HTC Dream	3.2"	320×480	181
Sony Ericsson W880i	1.8"	240×320	222
Nokia N80	2.1"	352×416	256

The second approach is to use a great pixel density, or pixels per inch. We often refer to screens by either their actual physical dimensions ("I have a 15.4-inch laptop screen") or their pixel dimensions, or resolution ("The resolution of my laptop is 1440×900 pixels"). The pixel density is determined by dividing the width of the display area in pixels by the width of the display area in inches. So the pixel density for my 15.4-inch laptop would be 110 PPI. In comparison, a 1080p HD television has a PPI of 52. As this applies to mobile devices, the higher the density of pixels, the sharper the screen appears to the naked eye. This guideline especially applies to type, meaning that as text is antialiased on a screen with a high density of tiny pixels, the glyph appears sharper to the eye. Some mobile

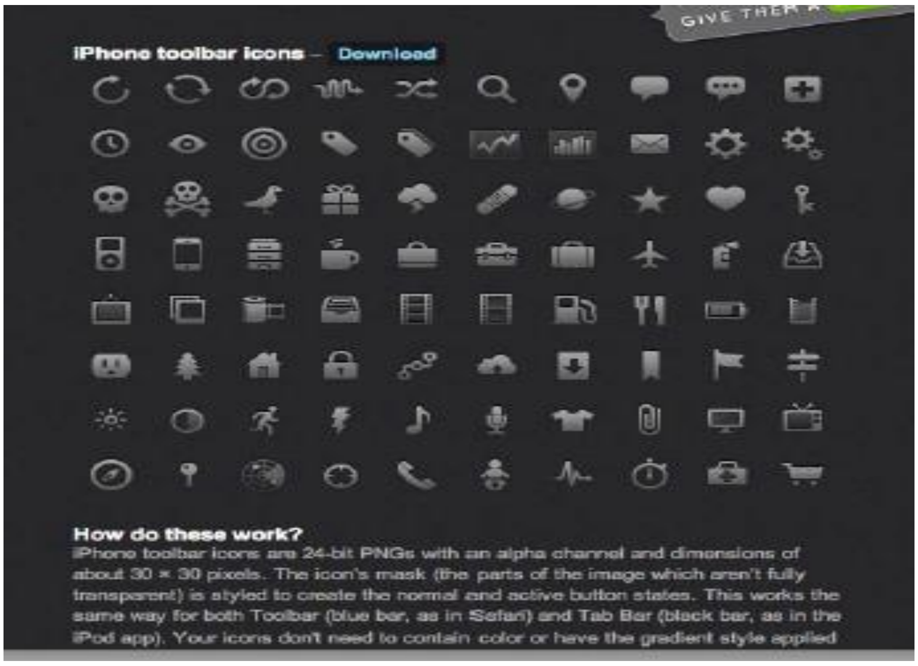
screens have both a high PPI and subpixel technology, though these are unnecessary together.
Fortunately, today’s mobile devices have a few more options than a single typeface, but the options are still fairly limited. Coming from web design, where we have a dozen or so type options, the limited choices available in mobile design won’t come as a big surprise. Essentially, you have a few variations of serif, sans-serif, and monospace fonts, and depending on the platform, maybe a few custom fonts.
In researching this book, I scoured the Web and tapped my mobile community resources to find a list of the typefaces that are included in each of the major device platforms, but I could only come up with a few—nothing close to a complete list. This goes to show how far behind mobile typography is, that designers don’t even have a basic list to work from.
Therefore, when creating mobile designs for either web or native experiences, my advice is to stick with either the default device font, or web-safe fonts—your basic serif variants like Times New Roman and Georgia or sans-serif typefaces like Helvetica, Arial, or Verdana.
Font replacement: The ability to use typefaces that are not already loaded on the device varies from model to model and your chosen platform. Some device APIs will allow you to load a typeface into your native application. Some mobile web browsers support various forms of font replacement; the two most common are sIFR and Cufon. sIFR uses Flash to replace HTML text with a Flash representation of the text, but the device of course has to support Flash. Cufon uses JavaScript and the canvas element draws the glyphs in the browser, but the device of course needs to support both JavaScript and the canvas element.
In addition, the @font-face CSS rule allows for a typeface file to be referenced and loaded into the browser, but a license for web use is usually not granted by type foundries.
Readability: The most important role of typography in mobile design is to provide the user with excellent readability, or the ability to clearly follow lines of text with the eye and not lose one’s place or become disoriented. This can be done by following these six simple rules:



ONE MORNING, when Gregor Samsa woke from troubled dreams, he found himself trans-formed in his bed into a horrible vermin. He lay on his armor-like back, and if he lifted his head a lit-tle he could see his brown belly,

Figure 8-16. Classics, an iPhone application designed with readability and typography in mind

Use a high-contrast typeface: Remember that mobile devices are usually used outside. Having a high-contrast typeface with regard to the background will increase visibility and readability.
Use the right typeface: The type of typeface you use tells the user what to expect. For example, a sans-serif font is common in navigation or compact areas, whereas serif typefaces come in handy for lengthy or dense content areas.
Provide decent leading (rhymes with "heading") or line spacing Mobile screens are often held 10–12" away from the eye, which can make tracking each line difficult. Increase the leading to avoid having the users lose their place. Leave space on the right and left of each line; don't crowd the screen. Most mobile frameworks give you full access to the screen, meaning that you normally need to provide some spacing between the right and left side of the screen's edge and your text—not much, typically about three to four character widths.
Generously utilize headings: Break the content up in the screen, using text-based headings to indicate to the user what is to come. Using different typefaces, color, and emphasis in headings can also help create a readable page.
Use short paragraphs: Like on the Web, keep paragraphs short, using no more than two to three sentences per paragraph.
Graphics:
The final design element is graphics, or the images that are used to establish or aid a visual experience. Graphics can be used to supplement the look and feel, or as content displayed inline with the text. The use of graphical icons in the iPhone experience helps to establish a visual language for the user to interact with to quickly categorize entries. On the S60 application, the wallet photo in the upper-right corner helps communicate the message of the application to the user.
Iconography: The most common form of graphics used in mobile design is icons. Iconography is useful to communicate ideas and actions to users in a constrained visual space. The challenge is making sure that the meaning of the icon is clear to the user. We can have some helpful icons that clearly communicate an idea and some perplexing icons that leave you scratching your head.



How do these work?
iPhone toolbar icons are 24-bit PNGs with an alpha channel and dimensions of about 30 × 30 pixels. The icon's mask (the parts of the image which aren't fully transparent) is styled to create the normal and active button states. This works the same way for both Toolbar (blue bar, as in Safari) and Tab Bar (black bar, as in the iPod app). Your icons don't need to contain color or have the gradient style applied

***Glyphish* provides free iPhone icons**

Photos and images: Photos and images are used to add meaning to content, often by showing a visual display of a concept, or to add meaning to a design. Using photos and images isn't as common in mobile design as you might think. Because images have a defined height and width, they need to be scaled to the appropriate device size, either by the server, using a content adaptation model, or using the resizing properties of the device. In the latter approach, this can have a cost in performance. Loading larger images takes longer and therefore costs the user more. Using graphics to add meaning to a design can be a useful visual, but you can encounter issues regarding how that image will display in a flexible UI- for example, when the device orientation is changed. In Figure 4.25, you can see how the pig graphic is designed to be positioned to the right regardless of the device orientation.

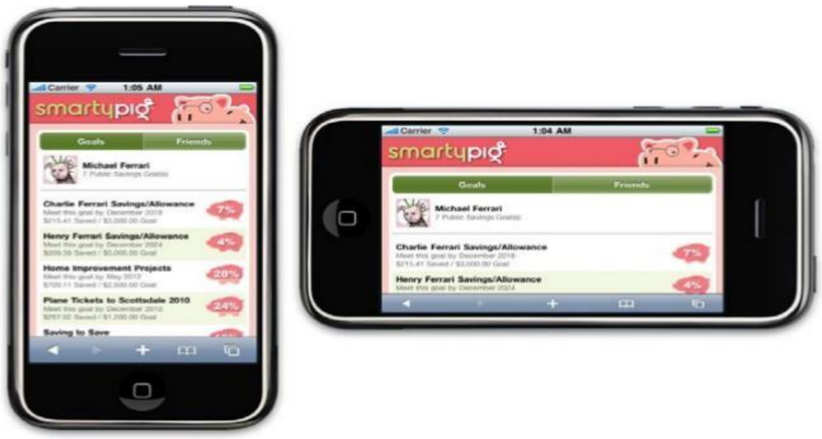


Figure 4.25. Using graphics in multiple device orientations

4.6 MOBILE DESIGN TOOLS

Mobile design requires understanding the design elements and specific tools. The closest thing to a common design tool is Adobe Photoshop, though each framework has a different method of implementing the design into the application. Some frameworks provide a complete interface toolkit, allowing designers or developers to simply piece together the interface, while others leave it to the designer to define from scratch.

In Table below, you can see each of the design tools and what interface toolkits are available for it.

Mobile framework	Design tool	Interface toolkits
Java ME	Photoshop, NetBeans	JavaFX, Capuchin
BREW	Photoshop, Flash	BREW UI Toolkit, uiOne, Flash
Flash Lite	Flash	Flash Lite
iPhone	Photoshop, Interface Builder	iPhone SDK
Android	Photoshop, XML-based themes	Android SDK
Palm webOS	Photoshop, HTML, CSS, and JavaScript	Mojo SDK
Mobile web	Photoshop, HTML, CSS, and	W3C Mobile Web Best

	JavaScript	Practices
Mobile widgets	Photoshop, HTML, CSS, and JavaScript	Opera Widget SDK, Nokia Web Runtime
Mobile web apps	Photoshop, HTML, CSS, and JavaScript	iUI, jQTouch, W3C Mobile Web App Best Practices

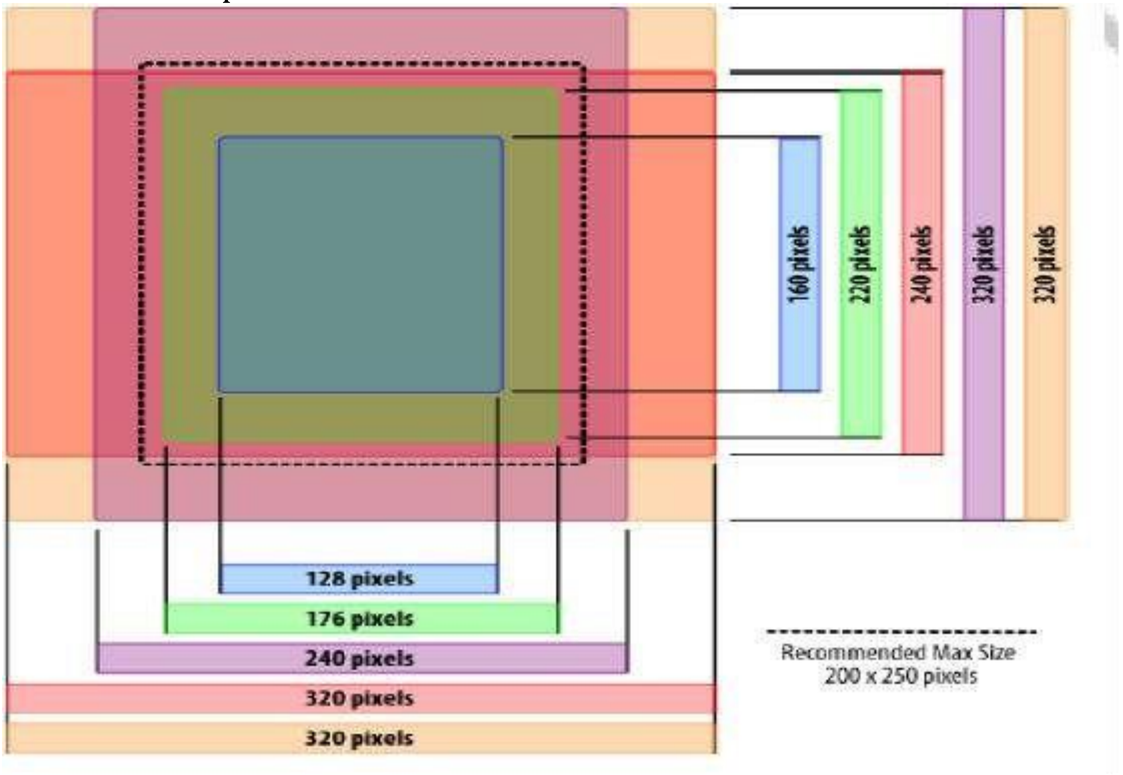
1. DESIGNING FOR THE RIGHT DEVICE

"What device suits this design best?
What market niche would appreciate it most?
What devices are the most popular within that niche?"

This knowledge will helps to develop porting and/or adaptation strategy, the most expensive and riskiest part of the mobile application.

2. DESIGNING FOR DIFFERENT SCREEN SIZES

Mobile devices come in all shapes and sizes. Choice is great for consumers, but bad for design. It can be incredibly difficult to create that best possible - experience for a surplus of different screen sizes.

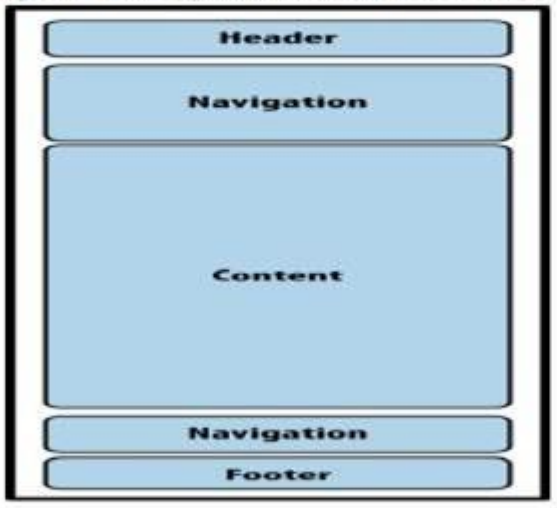


For example, your typical feature phone might only be 140 pixels wide, whereas your higher-end smartphone might be three to four times wider Landscape or portrait? Fixed width or fluid? Do you use one column or two? The vast majority

of mobile device screens share the same vertical or portrait orientation, even though they vary greatly in dimension.
With vertical. designs, the goal is to think of your design as a cascade of content from top to bottom, similar to a newspaper.

- The greatest challenge to creating a design that works well on multiple screen sizes is filling the width.
- For content-heavy sites and applications, the width of mobile devices is almost the perfect readability, presenting not too many words per line of text.
- The problem is when you have to present a number of tasks or actions.

Figure: The typical flow of information on mobile devices



CASE STUDY 1:

For a pioneering distributor of petroleum products and related services develop an official app which, leveraging on location services, helps find the nearby filling station to the user. User can click a station on the map and navigate to it.

In addition, the mobile app must give information about the products, services and facilities available at their filling stations. The upcoming events & ongoing promotions should also be available on the app to view and participate. Customers should be able to provide their feedback to help improve the products and services.

Discuss about the challenge and design to develop a mobile App.

Challenge

- The client app, upon registration, stores users' driver's license.
- To notify a user when his driver's license is about to expire.
- The challenge augmented is that the license expiry date tends to differ for user to user.
- The app upon registration needs to store vehicle and mileage data.
- However, with so much data on the fly, it is evident to think about storing the data.
- Filtering results to a region, location and sub-filtering them to services/features/facilities.

Design

- The colour palette was chosen such that it includes various shades of grey with light colours here and there.
- To make sure the users receive a notification when their driver's license is about to expire, Android Local Push Notifications—Alarm Manager to set the expiry notification was employed.
- To store the vehicle and mileage data, created the local SQLite DB and to manage complex filter, SOL query with app code was used.

- **Nearby stations**
- An app user can not only find a filling station near him, he can also navigate to its location and take a look at the services, products and facilities available at the station.

- **Mileage Calculator**
- The app calculates mileage of a vehicle based on the number of miles it travelled. It calculates number of miles it travelled and how much fuel it was refilled with an The app **filling station**.

- **License Expiry Notifications**
- The app notifies a user when the license is about to expire.

- **Rating and Feedback**
- Customers' feedback matters more than anything be it a grocery store or a filling station. The app has many station and each time an app user visits one, the app prompts him to rate the experience.

Results

- The app reduced time and fuel people waste while searching for a fuel station. The app users became safer and more considerate drivers. Driver's License expiry notification assured they renew their licenses on time.

CASE STUDY 2:

The client wants a mobile application to acts as a platform for its users to enroll themselves and their friends in the training center. The Training centerhas a total of six branches that provides a variety of courses and training to their students.

The application is bilingual and is available in English and Hindi. This application has many things to offer to its users. It gives a comprehensive information about all the courses whether they are ongoing or upcoming. Any user can avail these courses by making the payment through the app itself.

Discuss about the challenges and solution for developing the Mobile APP.

Challenges

The client is a reputed training institute. It's well-known for the variety of courses and quality of teaching. However, they were facing problems in reaching out to more students. They wanted to develop a mobile application that can work like a platform to enroll in their institute and also to guide their students.

Solution :

- To overcome the challenges of the client an app has to be developed to fulfil all the clients' requirements.
- With this application, all their users can see the number of different courses, its eligibility, timings, instructor, and many more.
- The app also provides all the guidance to the students related to the institute.
- **Explore courses with various filters**
- The institute has many courses and that's why this feature comes in handy as it allows its users to search for a course by using different filters such as category, age, location, gender, start date, and many more.
- **Bilingual**
- The client's institute is a reputed one and they wanted to attract students of all demographics. That's why we need to develop an app in both English and Hindi language.
- **Online enrollment& payment**
- This feature allows the user to enroll for various courses of their choices. Along with that they can also make payments online via the mobile app.
- **Gallery**
- Users can always have a look at the gallery that features pictures from all six branches of the training institute.
- **Notify me**
- This feature is a crucial one as it lets its user to set a reminder to any of the upcoming course. Whenever, the date of that course approaches near, the app reminds the user to enroll for it.
- **News**
- This feature shows all the news related to academics. It also shows the details about examination taken for various vacancies.

Result

- This app will turn out to be a huge success for clients. It will play a pivotal role in attracting many new students into the institute.

CASE STUDY 3:

With medical technologies taking huge leaps every year, most of our health-related woes are taken care of. Doctors are successfully treating some conditions which were once incurable. However, with the advancement, the number of tasks involving the medical procedure has also increased to an overwhelming level.

An app that helps to manage all those procedures and tasks in an easy and systematic manner need to be developed. This application brings together all the components of health report which includes the medical history, symptoms, medication, appointments, immunizations, allergies, and fitness records.

Discuss about the challenges and solution for developing the APP

Challenges

- First need to fetch the thehealthkit& Fitbit data on the application.
- Integration of both will be a challenge as it would increase the complexity of the app.
- Moreover, to fetch data in real-time from the healthkit& Fitbit to the application will be a tough job.
- Other challenge is about the medicine reminders. Client would need this feature in which the user would get notifications for his/her medical dose.
- Apart from that, need to provide encryption and decryption of data which would ensure foolproof privacy and security of user's data.
- Need to provide multiple languages in the app.

Solution:

- To solve the first challenge can useHealthkit's& Fitbits' API to fetch all the data to the app.
- For the second challenge which was of medicine reminders, can synchronize with the calendars. Can add reminders according to the frequency of the dosage.
- For encryption and decryption, can use AES encryption algorithm that works parallelly across every platform such as iOS, Android, and PHP.
- To make this app available in all the languages can create a master in which all the language translations for all the modules can be stored.
- **Record symptoms**
- In this feature the user can record all his/her symptoms with details such as date of first appearance, severity, and other miscellaneous information.
- **Immunizations**
- The user can record history of all the vaccines, any other immunizations taken in a single app which will give the user and the doctor a clear idea about your immunization history.
- **Chronic conditions**
- This feature allows the user to record all the necessary details of his/her chronic disease which would help the doctor to take further action rapidly.
- **Allergies**

- In this section the user can fill all the details of the allergies if he/she has any. This will prevent cases in which doctor prescribes medication to which the patient is allergic, causing harm.
- **Fitness data**
- This is a crucial feature in which all the data from fitbit or healthkit is fetched to the app. Now the user can monitor his/her fitness stats and manage medication simultaneously**.**
- **Medicine dosage**
- The user can enter all the details such as their name, amount of dosage to be taken, and the frequency of a dosage of all the medicines that the user have been prescribed with**.**
- **Medical appointments synced with calendar**
- This is a critical feature as the user can sync all the appointments with the calendar. User will receive a reminder whenever the user have a medical appointment fixed that day.
- **Voice commands**
- This feature is useful for elderly users who may face problems in typing. They can simply give voice commands to access functions and to write notes.
- **Share PDFs, video, and audio files with the doctor**
- This feature gives the user a golden opportunity to present the medical history with a great amount of details to the doctor. In this feature the user can include PDFs, videos, and audio files forthe doctor.
- **Create multiple dependent accounts**
- The user don't need to create separate accounts for each member of his family. By this feature , the user can create and manage multiples account of his family members on the same application

Result:

- The application will get a tremendous amount of appreciation from all; especially, from those who had to manage a large number of medical tasks for themselves or their family members