Mobile Solutions for Your Library

Abstract

This chapter of Libraries and Mobile Services *puts the information from the previous chapters into a library-specific context. By examining what different libraries have already done to provide mobile services and providing best practices and suggestions for future implementation, librarians gain a foundation for implementing or expanding services in their own facilities.*

So, if the preceding chapters have been convincing, and you now understand that your library needs to begin strategizing for mobile access and delivery of services to mobile users, how do you start?

Become a Mobile-Only User

If you already have a mobile device of recent vintage, commit to using it as your primary device for accessing your library's site and systems. If you are an employee of an academic or special library or other library where you are not also a frequent patron, consider becoming a mobile-only user of your local public library.

Give it a week or two. Document the experience. Use your website, OPAC, licensed resources, reservation forms, chat service, guides, the works. Don't limit yourself to those systems created by your library. If your users access it through your website, you bear some responsibility for the user experience, whether it's within your control or not.

Keep track of those resources or systems that automatically detect your mobile device and either notify you of a mobile interface or direct you there automatically. You will likely find some instances where you hit dead ends or where you're unable to access files, pages, or systems. Note especially when user-facing interfaces or systems are challenging or impossible to navigate. Are there interfaces in which you consistently found yourself inadvertently hitting the wrong links or buttons? Systems that appear to serve up files that your device is unable to load? How much longer does it take you to complete tasks as compared to your usual workflow?

Figure 23
The University of Minnesota Libraries' website viewed in Mobile Safari.

As discussed above, e-book readers are available for nearly every smartphone platform, and most of the applications are free to install. Commit to using one of these tools to read your next book. Choose one of the many free public-domain titles that are available in these applications. Or take the plunge in the name of research and buy an e-book from the Kindle, iBooks, or Nook store. If your library subscribes to OverDrive (figure 24) or another e-book platform, try it, not as a library staff member, but as a reader.

This may be a humbling or frustrating experience. It will be eye-opening. It may also have very positive aspects. Note if there are situations where your mobility allows you to accomplish tasks that you couldn't if you needed to be seated at your regular workstation.

Figure 24
OverDrive, an e-book app popular with libraries.

Did you see another passenger on the bus or train reading a book that you'd been meaning to request? Were you able to request it at that very moment from your mobile device rather than hoping to remember once you got to your desk? Were you able to read on your mobile device long into the night without disturbing your partner with a bedside light?

Keep track of all your observations, both positive and negative, and share them with your colleagues. Encourage them to do the same. You will quickly be able to develop a list of areas where you should improve your mobile user experience.

Device Focus Group

Very few of us own or carry more than one current-generation mobile device, even gadget obsessives like your author (though once there's a contract-free Android equivalent of the iPod Touch, I expect that to change). The mobile device landscape is diverse, and becoming more so each day as new and worthy competitors to the established platforms enter the stage. So while our individual experiences with a single mobile device are instructive in creating a mobile strategy, they will not represent the diversity of our user base.

Canvass your staff, users, friends, family, whatever it takes, and find a willing group of people who carry a representative sample of the most popular smartphones and operating systems on the market. If I were convening such a group today, I'd hope to have a user each with an iPhone, a Palm Pre or Pixi, a device running the current and at least one past version of Android (such as 2.1 and 2.2), and user each for a recent BlackBerry device running the 6.0 version of the operating system and one with an earlier version. Windows Phone 7 devices may take some time to penetrate the market, but if you can find someone who uses one, so much the better. If possible, convene the users together on-site with a few of your public services, systems, and IT staff. It may be useful to use an opaque projector to display users' handsets on a screen so not everyone has to hover over the users' shoulders. Appoint at least one person to take detailed notes.

Ideally your focus group would consist of more users than staff, in which case you can begin by asking about their typical usage patterns for your services and systems. Do they typically access your systems from their home or work? Or from workstations in the library? Do they use your licensed databases, indexes, or e-books? If so, which ones? Do they often reserve books or other materials online for pickup in the library? Perhaps most importantly, are they already using their mobile device to interact with the library? If so, how?

Once you've established a list of your users' typical tasks, ask each user to walk your staff through any library tasks for which he or she is already typically using a mobile device. As is often the case, expect to see some usage patterns that you don't expect. Rarely do our preconceived notions of our users' methods match the reality. Next, ask users to step through typical nonmobile library tasks. Can they easily browse your collection? Access their account information?

With each of these sites, services, and tools, document what works, how well it works, and what doesn't work. You'll likely find that with ample zooming, tapping, and fiddling with tiny controls, your mobile users can accomplish a great deal with their devices. You will also likely find some tasks they can't accomplish. Finally, expect to find some tasks that, though possible, are challenging or time-consuming enough in the mobile environment that your users are unlikely to attempt them again.

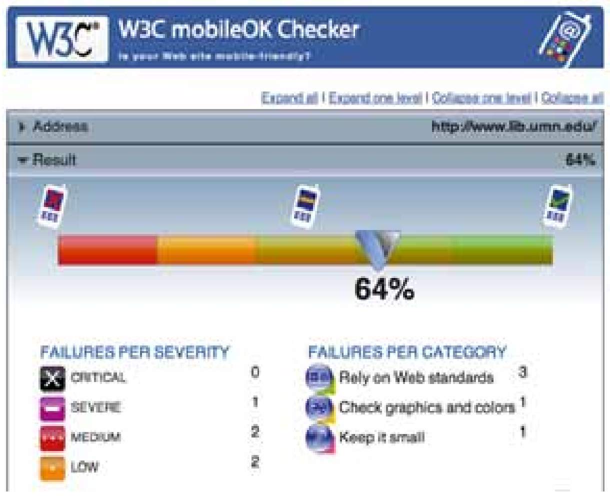
This process should give you a good idea of where your library currently stands: which of your tools and services are usable, which may be frustrating, and which are downright broken for mobile users on various platforms. Once you have this list to start from, you can undertake one of four methods for beginning to develop your mobile service:

- Revamp your website for mobile-friendliness.
- Develop a mobile website.
- Develop mobile web applications (by which I mean applications that run within a mobile device's web browser).
- Develop native mobile applications.

We'll explore these options in the next sections.

Ensure Mobile-Friendliness in Your Current Site

In your audit of your website and other tools using various mobile devices, you likely found that the vast majority of tasks can be accomplished by mobile users, provided they have requisite patience. You also likely found some situations where users are stymied by code or multimedia that was designed for desktop users without regard for alternative platforms. Even if you have limited staff resources, there are some steps you can take to improve your existing site and make it more mobile-friendly.


W3C mobileOK Checker
Is your Web site mobile-friendly?
Expand all | Expand one level | Collapse one level | Collapse all

Address http://www.lib.umn.edu/

Result 64%

64%

FAILURES PER SEVERITY		FAILURES PER CATEGORY	
CRITICAL	0	Rely on Web standards	3
SEVERE	1	Check graphics and colors	1
MEDIUM	2	Keep it small	1
LOW	2		

Figure 25
The W3C mobileOK Checker can help you verify the functionality of your application.

Validate

First things first. Even though you have anecdotal evidence of what works and what doesn't, you ought to run your site through the World Wide Web Consortium (W3C) mobileOK Checker (figure 25). This Web-based tool will evaluate the extent to which your website adheres to the W3C's Mobile Web Best Practices, a set of standards for mobile-friendliness of web documents.

W3C mobileOK Checker
http://validator.w3.org/mobile

W3C's Mobile Web Best Practices 1.0
www.w3.org/TR/mobile-bp

If your site is like most, the mobileOK Checker will provide you with a list of "failures" ranked by severity (Critical, Severe, Medium, and Low) and category. The failure categories correspond directly to specific items in the Mobile Web Best Practices document, and links to the relevant best practices are provided in line with the errors. Failure categories include "Rely on web standards," "Keep it small" (which refers to the total file size of your site), and "Stay away from known hazards," among others.

You will likely find that some of the failures reported by the mobileOK Checker are easily fixed by updating your page headers to properly reflect your site's character encoding and doctype. Sadly, while it's good to get these cleared up in order to avoid throwing errors in users' browsers, doctype and character encoding are unlikely to be the cause of your site failing for mobile users.

Fixes for Mobile-Friendliness

Give Your Site Some Space

I wear size XL gloves, XXL in some brands. Despite my gargantuan fingers, I've become quite facile with the virtual keyboards shipping on today's mobile devices, largely, I expect, due to the very clever error-correction software onboard in mobile operating systems. I'm not so lucky when it comes to using the Web on mobile devices. When I use sites that are not designed for mobile, I typically find myself tapping, pinching, zooming, and ultimately using just the very tip of my finger to hit links and buttons that were clearly designed for mouse and keyboard or for elven fingers. Here's a case where mobile-friendly design is probably just good design. Pad your navigation elements. Limit the number and proximity of links within text. Simplify.

Avoid Mouseovers

No mouse, no mouseovers. Makes sense, right? Some mobile web browsers will adjust by interpreting a tap on an element as a mouseover when one is specified in the code. At best, this means that flyout menus require twice as many taps for mobile users as they would for mouse users: tap once to expand the menu, tap again to select an element or expand a submenu (figure 26).

In the case of the menu in figure 26, selecting Bottle Cappers on an iPhone would require tapping once on Equipment to expand the Equipment menu, once on Bottling to expand the Bottling submenu, once on Bottle Cappers to invoke the mouseover change to the darker color, and then once more to follow the link. Or so I hear. The browser in webOS interprets a tap only as a click, meaning that it's impossible to expand the flyout menus. Tap on Equipment in your webOS browser, and you'll go to the full Equipment page, though you may see the menu flash on the screen briefly. Android's browser splits the difference, and a tap on Equipment both follows the link to the Equipment page and also opens the Equipment menu, with no clear means to close it.

Of course, these flyout menus are ubiquitous for a reason: they're a really easy way to provide quick access deep into your site. If you feel your site's navigation requires this kind of nesting within a page, try not to double up on functions within a single button. Use a button or label to expand a menu or as a link to a page, not both.

Mobile-Friendly Multimedia

A note to sensitive fanboys and fangirls: I'm about to use the F word. Flash. Much has been made of its presence or lack thereof on mobile devices. At present, Apple has shipped tens of millions of mobile devices that do not, and likely will not ever, run Flash. Even Android doesn't ship with Flash enabled by default. Avoid Flash.

Fortunately, the vast majority of online video services are seamlessly serving up non-Flash versions of their content to mobile devices and other Flash-free platforms. This means that if your library's tutorial videos or screencasts are already up on YouTube, Vimeo, or another major video provider, you may not need to worry about converting them.

If you are encoding and hosting your own Flash video, consider re-encoding from your source files using the H.264 codec, which has as close to universal device support as is currently possible. At the bare minimum, provide links to mobile-friendly files alongside any embedded video. Better yet, use a fully HTML5-compliant embed of an H.264 video.

The use of Flash for navigation or layout elements in a way that degrades appropriately for mobile devices or other platforms without Flash has, to put it mildly, a high degree of difficulty. Unless you've got some serious Flash development talent on staff or on retainer, it's best avoided.

The Nuclear Option

If we're honest with ourselves, an assessment of any library home pages will reveal links, sections, images, or features that are not strictly utilitarian. If you've got extraneous content on your site and are seeking to simplify for mobile use without redesigning, Cascading Style Sheets offer a simple way to nuke chunks of your pages. Similar to the method for designating a separate style sheet for print views of your pages, you can specify a mobile style sheet using the "media" attribute of the "link" tag when embedding your CSS like so:

```
<link rel="stylesheet" href="your_mobile_
stylesheet.css" media="handheld, only screen
and (max-device-width: 480px)" />
```

The above use of the "media" attribute indicates that this style sheet is to be used when a browser accessing the page is designed to use the "handheld" style sheet or has a screen of sufficiently low resolution. This technique will effectively capture most mobile devices, although newer high-resolution displays may require more targeted coding.

Once you've pointed your mobile users to their own style sheet, simply identify the extraneous bits of your pages that can effectively be removed. Give those divs the CSS "display:none" property, and they won't appear for anyone using a mobile device. This technique is crude, but if your home page is dominated by a Flash slideshow that your director is in love with, it may be your easiest option for accommodating mobile users without much custom coding.

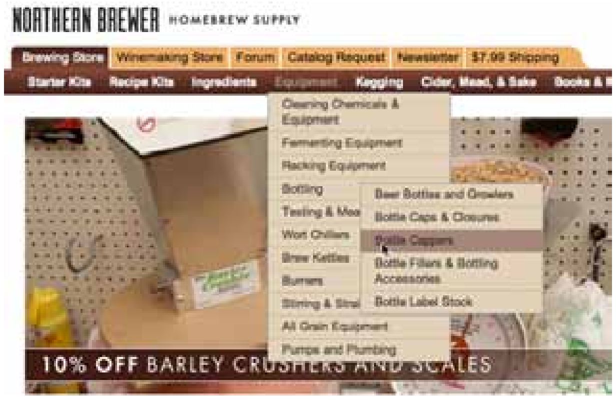


Figure 26
While mouseovers are a good feature on a standard website, they are problematic on mobile devices.

Develop a Mobile Website

If you have a little more time and resources to devote to your library's mobile strategy, consider developing a basic mobile website. It doesn't need to be anything fancy in order for it to be useful. By repurposing some basic content and linking to third-party mobile tools, you can likely cobble together a site that will be just what your users need when they want to access your library info on the go.

Designing and Developing Your Mobile Site

You can build mobile webpages just as you would any other webpage, using nothing more than a text editor. You will have the best results if you rely on web standards and build your pages using current versions of HTML, XHTML, and CSS. If you're familiar with jQuery or other JavaScript frameworks, there are a number of options for creating robust interactive sites with JavaScript, including the recently released jQuery Mobile.

jQuery Mobile: Demos and Documentation
http://jquerymobile.com/demos/1.0a1

Many of the leading mobile device manufacturers offer developer documentation on best practices for web applications. If you want a quick overview of design principles that apply to many mobile devices, I suggest HP/Palm's documentation "Developing Web Content for the HP webOS Platform." For an exhaustive reference on best practices for iOS devices, consult Apple's iOS Human Interface Guidelines.

Developing Web Content for the HP webOS Platform
http://developer.palm.com/index.php?option=com_content&view=article&id=1574&Itemid=53

iOS Human Interface Guidelines
https://developer.apple.com/library/safari/#documentation/InternetWeb/Conceptual/iPhoneWebAppHIG/Introduction/Introduction.html

In perusing Apple, HP, Android, BlackBerry, and Windows Phone 7 developer documentation, a few basic best practices become clear.

320 x 480 Layout

While the iPhone 4 and some newer Android handsets have a higher-resolution screen, sticking to 320 x 480 (figure 27) will ensure that your site is fully functional with the installed base of older smartphone models.

Viewport

Define a viewport in your site's meta tags that matches your mobile site's layout. The viewport sets the initial level of zoom for your site in the mobile browser. Smartphone web browsers are packed with features intended to make the nonmobile-optimized Web usable, and among them is a default viewport of nearly 1,000 pixels, designed to show the entirety of a website's width on initial load (figure 27). If you've designed a 320 x 480 pixel site and fail to set a matching viewport, your site will appear to have been shrunken to a third its actual size when initially loaded.

15-Point Text

The specific recommendations vary by manufacturer, but the average seems to hover around 15 points as the minimum size for readable text on a mobile webpage.

The screen is smaller than you think: If you're using an emulator on a standard computer monitor for development, you're seeing your designs at a much lower pixel density than on a mobile device.

Redirect Sparingly

Using packages like WURFL (the Wireless Universal Resource File), it's possible to detect specific platform/browser combinations on your web server and redirect them to your mobile site. Keep in mind that many of today's mobile devices can render nonmobile sites, and your users may prefer to stick with the site they're accustomed to. Consider redirecting users to an intermediary page where you give them the option of going to the mobile site or the main site (figure 28). Set a cookie to remember their preference.

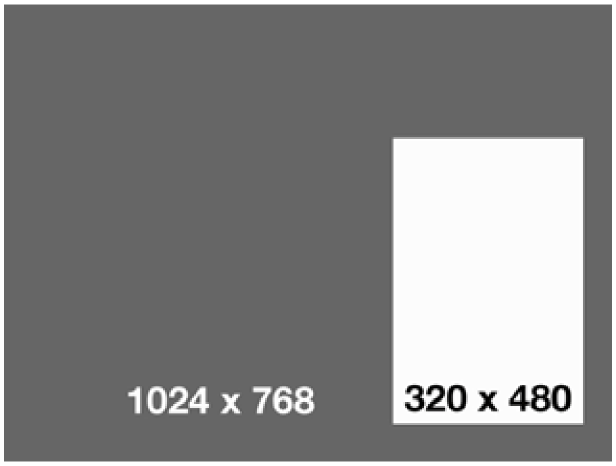


Figure 27
Viewport variation between the desktop and mobile browsers.

WURFL
http://wurfl.sourceforge.net

Link to Your Main Site

Always provide a link to your nonmobile site. Not all of the functions of your main site may be available on your mobile site. Provide a path back to your main site in the footer of your mobile home page.

Local Content

Our experience with mobile web development at the University of Minnesota Libraries echoes what I've heard anecdotally from a number of colleagues at other institutions: the pages we use to serve up library hours information to mobile users are our most popular mobile pages by far.

I mention hours not just because I think it's important to make your hours easily accessible to your mobile users, but because it's instructive for considering the kind of information you serve mobile users and the contexts in which they're accessing said info. Users opt for accessing our site via mobile device when they are seeking small, concrete bits of task-specific information. In a rush to make it to the library? If you're like me, you're far more likely to risk tripping over a curb while finding hours or location information on a mobile device en route than you are to take the time to find the information using a laptop or desktop computer.

To quote from an earlier version of Apple's Human

Interface Guidelines:

> iPhone users are accepting of, and even anticipating, an experience different from the one they are accustomed to on a desktop computer, a laptop, or even a mobile phone. Although this affords you a certain latitude for experimentation, be aware that iPhone users are likely to be even less tolerant of sluggish performance and a complicated user interface than they are when it comes to software running in a computer.[1]

So, even though the University of Minnesota Libraries offer a fairly robust custom mobile search interface for our catalog and quite a few mobile databases, it's the hours and basic contact and location information that take the top slots for mobile users (figure 29). It's easy to fret about the complexity of creating mobile skins for our OPACs and federated search tools, but when it comes down to it, the most important information is relatively flat, text-based, and under our local control.

Vendor and Third-Party Mobile Interfaces

The number of vendors and aggregators providing mobile interfaces for their products has increased dramatically in the past couple of years. Whether you know it or not, your library likely already subscribes to one or more databases that already have mobile interfaces. Talk to your vendor reps and begin keeping track of which resources provide mobile web interfaces. You may be able to provide your mobile users a great deal of functionality simply by taking advantage of the work your vendors have already done for you.

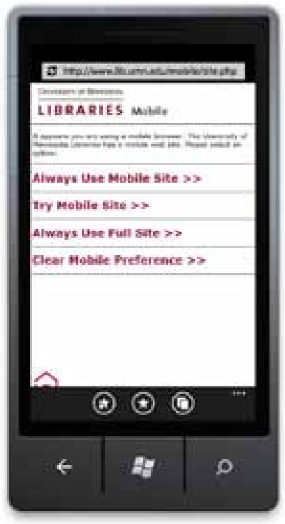
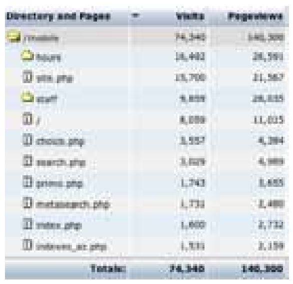


Figure 28
The University of Minnesota Libraries' mobile site allows users to set a preference for the mobile site or the full site.

Don't have the time, money or expertise to develop or purchase a mobile front end for your catalog? Consider linking to WorldCat Mobile (figure 30), where your users can answer the type of question they're likely to have on the go: does a resource exist that meets my need?



Directory and Pages	Visits	Pageviews
/mobile	74,340	140,300
hours	16,482	28,591
site.php	15,700	21,567
staff	9,859	28,035
/	8,059	11,015
choices.php	3,557	4,394
search.php	3,029	4,989
primo.php	1,743	3,655
metasearch.php	1,731	2,480
index.php	1,600	2,732
indexes_az.php	1,531	2,159
Totals:	74,340	140,300

Figure 29
Stats from University of Minnesota's mobile site.

WorldCat Mobile Web Beta
www.worldcatmobile.org

If you're an Ebsco subscriber, you can direct your users to mobile versions of your Ebsco databases (figure 31), each of which will provide mobile-optimized access all the way to full-text PDFs, which should be viewable on most current smartphones.

PubMed has offered a spartan, but thorough, mobile interface for years, and it is quite usable on today's smartphones.

If your library uses the QuestionPoint or LibraryH31p for chat reference, each offers a version of its chat widget that is accessible by mobile devices.

Develop Mobile Web Applications

If you have access to sufficient developer expertise, you may consider adding levels of functionality to your mobile website by creating custom interfaces for your local systems, such as your OPAC, federated search, ILL, and circulation/account system. Increasingly these systems offer application programming interfaces (APIs) that allow for programmatic interaction with other applications. Such APIs in an OPAC or ILS might, for instance, allow you to take the input from a simple web form and pass it to the OPAC as a search query, then receive the results of that search back as a data stream that you can skin appropriately for your mobile web application.

You may be able to craft a similarly mobile-optimized interface for a link resolver, allowing users to search or browse for electronic journal titles. At the University of Minnesota, our developer John Barneson realized that while publishers and other vendors didn't consistently offer mobile versions of their sites, our federated search tool had a robust API, allowing him to create a mobile-optimized search interface not only for traditional federated searching, but for searches targeted to single databases. Using the API meant that we could, in one fell swoop, have a mobile search interface for any of our databases that were compatible with our federated search tool (figure 32).

When libraries begin to devote the time and resources necessary to create these more complex mobile tools, it's important to spend some time thinking about not just what is possible, but what's truly appropriate to present to mobile users. It is possible to direct users to Ebsco's mobile interface, or even to bring

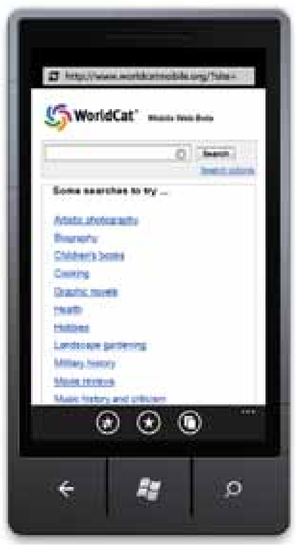


Figure 30
WorldCat's mobile interface.

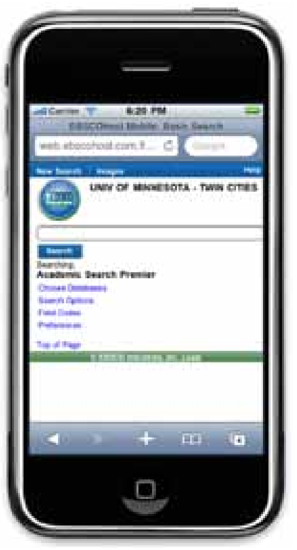

EBSCOhost Mobile: Basic Search
UNIV OF MINNESOTA - TWIN CITIES
Search
Academic Search Premier

Figure 31
Ebsco's mobile app.

them directly into the database via a mobile federated search interface. And once there, it is possible, on the vast majority of smartphones, to open up PDF files containing the full text of articles. However, the prospect of reading these PDFs on most mobile devices is still quite unappealing. The layout of most PDFs has been optimized for 8½-by-11-inch paper in portrait orientation, and, let's be honest, they're a pain to read on an average laptop screen, much less a mobile screen that may be as small as one tenth the size.

So, reading full-text PDFs is possible, but not necessarily appropriate. In considering common contexts of mobile device use, it's more appropriate to think of the user not in the mindset of concerted researching, but instead in brief moments of verification. Rather than working to support complete digestion of an information object, we should optimize our web applications around discovery and, most importantly, marking or holding for later retrieval. The appropriate calls to action in a mobile library web application should be "bookmark this," "request this item," "e-mail me these records," and so on.

There is one gee-whiz mobile web application feature that I'd like to see more libraries implement. Most smartphone browsers support the draft HTML5 Geolocation API, which allows the user to grant webpages access to the phone's location information. Likewise, many licensed electronic resource agreements allow for free public access on site in the library. How cool would it be to create your library web application such that users could be authenticated into your electronic resources on their mobile devices by proving that they were in your library via geolocation? Very cool is the answer.

W3C's Geolocation API Specification
http://dev.w3.org/geo/api/spec-source.html

Develop Native Mobile Apps (or Don't)

As mentioned earlier in this report, the iOS App Store for iPhone, iPod Touch, and iPad has been a juggernaut. In the two and a half years since the App Store opened its virtual doors within Apple's iTunes media management and syncing application, more than 250,000 applications have become available, and have been downloaded over 6.5 billion times. Developers of paid iOS applications split the sale price 70/30 with Apple, and by mid-2010 the 70 percent that Apple pays developers had topped \$1 billion.[2]

The Android Marketplace has lagged behind the iOS App Store in the number of applications available for download, the number of applications downloaded by users, and the profits realized by developers. However, as Android adoption nears or surpasses that of iOS, the Android development scene has begun to blossom.

Apps have truly become the watchword of this new mobile renaissance. But you probably shouldn't build one.

Let me explain. There are three reasons why you might want to build a native mobile app:

1. Profit: It's clear that it's possible to make money building software for mobile devices, especially for iOS.
2. Native device functionality: If you have an idea for an application that requires access to the camera, the compass, the tilt sensor, you won't be able to do that with a web app.
3. Exposure: You may assume that simply having an app will burnish your reputation among the cutting edge and provide opportunities for press releases.

So, to examine each of these reasons. First, if you manage to figure out a way for your library to turn a profit with a native app other than by jamming it full of advertising, call me. With some notable exceptions,

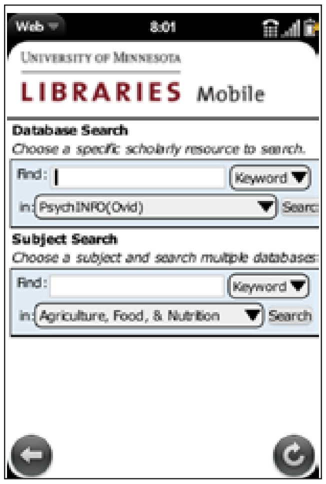


Figure 32
Searching a database.

we're not in the business of making a profit, nor for charging for our services. So, unless you have a very unusual library or some very unusual collections that would lend themselves both to mobile applications and to monetization, it's unlikely that profit is an appropriate motivation for building an app.

Implementing native device functionality can be very, very cool. Apps like RedLaser (owned by eBay) use the iPhone or iPod Touch camera to scan UPC barcodes so you can compare prices on items across various online resellers. They've also partnered with WorldCat so that when you scan a book you can see if it's available in a local library. RedLaser even provides an SDK for its barcode scanning function, meaning a library could craft its own app using this technology. Functionality like this is simply not available through anything other than a native app.

There's no doubt that having mobile apps provides an opportunity to talk up the cutting-edge developments at your library. However, it's no longer the case that simply having an app will drive users to your library. Between the various smartphone platforms, there are nearly half a million applications available, and a library application with local or regional reach is unlikely to make it into the "top download" charts that drive massive adoption.

There is one overarching reason why you might not want to build a native mobile app: If you want to do it well, it's expensive.

Craig Hockenberry, the developer of the popular iOS Twitter client Twitterrific recently responded online to a programmer who speculated that the app likely took 160 hours of development time and 40 hours of graphic design time (plus another month for testing). Hockenberry implied that estimate must have been generated under the influence of illicit substances. A recent update to the program had in fact taken more than 1,100 developer hours and 225 designer hours. He estimated the total cost at nearly a quarter of a million dollars in time and expenses.[3]

Now, Twitterrific is a highly polished application, but at its core, it's simply a mobile-native interface for Twitter's API. It's not that much different conceptually from what a library's native app might look like, except that a library's app would likely be more complicated and involve creating an interface for a number of different systems' APIs. And Twitterrific is only for iOS. Developing for Android, BlackBerry, webOS, or Windows Phone 7 would require a complete re-creation of the code, as these platforms share almost no common elements.

It is certainly possible to create native applications on the cheap. There are some tools like PhoneGap that provide options for building a single application that can be deployed to multiple mobile platforms. If you already have a mobile website, it's possible to package it in an application that acts as a simplified web browser dedicated to your site. However, in doing so you're asking your users to take the additional steps of downloading and installing an application and are offering no additional functionality to justify their effort. Worse, unless you devote developer time to creating data that can be stored locally on the users' mobile device, you've created an application that won't work offline, one of the key differentiating factors of apps versus the Web.

PhoneGap
www.phonegap.com

Notes

1. Apple Developer Connection, "It's a Browser-Based World," 2008, www.interactivefish.com/apple/section1.php (accessed Jan. 4, 2011).
2. Apple, "Statement by Apple on App Store Review Guidelines" (press release), Sept. 9, 2010, Apple Press Release Library, www.apple.com/pr/library/2010/09/09statement.html (accessed Jan. 5, 2011).
3. Craig Hockenberry, answer posted Oct. 13, 2010, to "How Much Does It Cost to Develop an iPhone Application?" posted Oct. 16, 20008, by user27815, http://stackoverflow.com/questions/209170/how-much-does-it-cost-to-develop-an-iphone-application/3926493#3926493 (accessed Jan. 4, 2011).