Best Practices For Mobile App Development

Introduction

There have been many debates in the past regarding whether Mobile is the way to go forward in the future or not, rather one-sided debates though. And as expected and predicted by experts and the rest—Mobile phones have taken over the world technology scene by storm. It becomes imperative that to be counted among the people who are ready for adaptation in this ever-changing world of technology to move ahead, from floppy's to CDs, from Mainframes to PCs and now from Desktops to Mobile.

A big part of this transition is to make your products and services mobile ready, which means to have a presence on each and every leading mobile platform. And what better way than an app to dominate the mobile landscape and grow your product even further. In this whitepaper, we proceed ahead with the steps and best practices you may follow to achieve a great start on the most enterprising platform ever created.

Making an App : Best Practices

Consider Your audience

Before you take any time to build an app, consider your audience. What do you hope to achieve? How do you envision your audience using your app? These are important questions to consider up-front.

Check the App Stores

Despite how unique you might think your idea is, there's an excellent chance that someone might have already built it,

Many times people come up with a great idea for an app and start to brainstorm how to build it. There's only one problem. Despite how unique you might think your idea is, there's a chance that someone might have already built it, or something similar to it.

Invite potential users in Design process

One danger of any design process is working only with your team and not involving the end users at all. Then, when the design is done and is released to the public, some or many aspects of your design might not translate well to the real world. To avoid this problem, involve potential end users in the design process and use their feedback to make changes as necessary.

Create a Story Board

The storyboard is one of the most important aspects of the design process. This is where you lay out the complete functionality of your app on paper. If there are problems, you can resolve them at this stage. The storyboard allows you to plan out all aspects of the design, including future components, such as plug-ins.

Make the App easy to understand

The app should be easy to understand with descriptions to accompany graphics (if necessary) and additional instructions. One design flaw is relying too much on images to tell the tale.

Pegasus One

Avoid overuse of Graphics / Animations

Both graphics and animations can add a nice "Wow" factor to your app but there's a major downside – slow loading times which translate into a poor user experience. Whenever possible, either avoid the use of bitmaps or animations or limit their use to only essential features.

Consider little details like button size/color

When working with a mobile interface, you have a limited amount of space and some designers add too many buttons/icons. Another consideration is the size of the human fingertip. If the buttons/icons are too small, users could make errors with selecting the wrong one. Likewise, if there's not enough space between the buttons/icons, that can cause trouble as well. If in doubt, test your layouts and get feedback.

Create a consistent workflow

This translates into making sure the user experience remains the same on all platforms. This is an important consideration given that many users today own a multitude of different devices.

Test the design

With any design, this is the most important aspect. If you've been following the strategies listed in this article you'll be testing your app every step of the way. Still, it's important to test the finished product and not only once but several times with different users. If there are problems, fix them, then test the result again.

Whenever possible, either avoid the use of bitmaps or animations or limit their use

Best Practices: Design

This is no longer a single-device world. The new mobility comes in new form factors: tablets, TVs, phones, notebooks, smart watches, to name a few. By focusing on just the handset experience or on just one device at a time, designers are missing an opportunity to craft deeper connections with their end-users.

How do we help these devices work better together, rather than just coexist? The answer lies in "Distributed Experiences" — digital solutions that reach across the various devices within an ecosystem. Keeping that in mind, we should ensure to follow a defined path on our journey to mobile excellence, and it all begins with the right preparation for the future.

Prototyping

Amateur mobile developers usually jump into development phase as soon as they hit a million dollar idea without giving a decent second thought about how to go about that and making sure that the idea is actually good to look at and interact with.

1. Jumping straight into the design is the fastest way to slow down the project in the long run. Developers often tend to skip the "User Experience" part just to hit deadlines, or often because they are least bothered about it. Skipping UX design always results in delays down the road as questions arise and must be discussed. So be sure to have a proper plan laid out beforehand for the same. Also, the thin line between UI and UX must be extremely well understood. UI (User Interface) must distill down from the desired UX (User Experience) and not the other way round.

2. **Iterate:** As much as possible. This ensures the product is as close to final as possible, at every cycle of development.

3. **Less is More:** Keep these magic words in mind when designing your app. Instead of 10 buttons that do 10 tasks try to incorporate only three even if they perform just 9 tasks. It might cost you few quick functions, but it will benefit you a lot by enhancing your user experience a lot, making it easy to understand your app.

4. **Un-complicate:** Crafting visually appealing UI that remains intuitive and simple is more challenging than creating a complicated UI. So take time to hit the right strides at all times and avoid any elements that are unnecessary/avoidable and add to clutter. Great designs are all about taking away complexity rather than layering on more of it.

A great way of simplifying the user interface is to review every visual element and determine how it benefits the user. Avoid trendy elements in design just because everyone is doing it.

5. **Design for thumbs:** While designing remember that almost every interaction with a modern mobile device is by a thumb or a finger. This calls for a simple

Pegasus One

uncluttered interface. The fewer the elements that are crammed together, the easier it is to navigate.

6. Readability: Readability is highly important and using minimum font sizes will help achieve this. But keep in mind not to make them small enough to be painful to read. Also, maintain a uniform look and feel across the app. This helps in navigation.

6. Perception of performance: Clearly the performance of an app directly affects the user experience. However, it is not always in your control. Network conditions, device specs etc contribute to the perceived performance of an app. So it is imperative to provide the user with some information about his actions that have been registered. Not notifying them of the same could give a wrong perception of the app being slow. Use spinners and loading indicators when you know the task will take a while to complete. This keeps the user engaged and interested in what's going on.

Takeaways

Never rush UX/UI design. It is one of the most important aspects of great app development and can also be the difference between a good app and a great app.

⇒ SIMPLIFY, DECLUTTER

⇒ FOLLOW BEST PRACTICES AND DESIGN GUIDELINES

⇒ GIVE USERS A PERCEPTION OF PERFORMANCE, EVEN WHEN OUTSIDE YOUR CONTROL

Best Practices: Architecture

When developing mobile applications, there are a number of key challenges where architecture and design are fundamentally different from that of a typical enterprise application. Careful consideration should be given to these mobile architecture issues early in the development process in order to mitigate the downstream impact of poor architectural decisions. While some of these best practices also make sense for the development of non-mobile applications, many will become more readily apparent when developing on a mobile platform. The five most important areas for consideration, which are detailed throughout this document, include performance, usability, data access, security, and connectivity.

Performance

While more readily apparent in the previous years of mobile development, the computing power available on mobile devices still lags behind desktop and server counterparts although it is catching up fast with quad cores and gigs of RAM thrown in every year. Still, devices feel sluggish and outdated with no time due to taxing and resource hogging apps. The quality of data connections available on a mobile device is often highly variable based on signal strength and is far inferior to broadband Internet access in most cases.

Often during rapid application development, performance considerations are ignored until the end of the project and optimized only when necessary. In mobile development, more consideration to performance constraints of the mobile device may need to be given upfront in the design process. Each platform has different code-level best practices for performance optimization depending upon the programming language and frameworks available on the platform. Some best practices, such as judicious use of memory and limits on the number of unnecessary objects created, however, can be applied across all platforms.

Care should especially be given to architectural decisions that can limit performance and are also difficult to change later in the development cycle, such as the design of web service APIs and data formats.

General best practices for the design of web service APIs for use in mobile development could be summed up as:

- Only retrieve data that the application needs
- Only retrieve data when the application needs it

These considerations stem mostly from the limited bandwidth available to mobile devices. If possible, APIs used by a mobile application should be designed to retrieve only the most relevant and useful information— excluding any extra data that is not used by the application. When designing APIs to communicate with mobile applications, one recommendation is to use a lightweight data format like JSON instead of more verbose format such as XML in order to make the best use of the limited bandwidth available to mobile devices.

The use of a lightweight format like JSON will conserve bandwidth, will allow results to be retrieved more quickly, and also will generally enable faster de-serialization of the data as it arrives on the mobile client.

Another important performance consideration on a mobile device is battery life. If an application is constantly polling a web service for updates or continually processing data in the background, the battery will be drained much more quickly. If architecturally feasible (and if the push notification capabilities exist on the mobile platform), the use of push notifications for providing data updates is recommended over periodic polling.

Usability

At the end of the day, usability is one of the key factors that will truly make or break user acceptance of an application. Each of the major mobile platform software vendors (Microsoft, Google, Apple) have released user - experience specifications and guidelines specific to their own platforms in an attempt to foster a consistent look and feel across all applications on their platforms—and if the guidelines are enforced by the vendor and followed by developers, then the payoff is absolutely realized. The user experience across applications on most of the major platforms is seamless—for example, on the more stringent iPhone and Windows Phone platforms, the navigation of menus and the look and feel of most applications (down to the fonts and color schemes) are almost identical. This allows users to learn quickly how to use a new application and instead focus on performing the task at hand, rather than "switching gears" between disparate experiences or puzzling over how to interact with a new application.

While each platform may have specific user interface (UI) guidelines, the challenges of mobile application usability are ubiquitous and many best practices can be applied across all platforms.

1. Consider the limited screen real estate. No longer do users have access to a 23-inch widescreen monitor to display every single piece of information at once. Only display the most relevant information and options on the screen.

2. Menus and UI screens should not be cluttered within rarely used options; rather they should be buried deeper within a settings screen or a submenu. Conversely, if a feature is used on a regular basis, consider assigning it to a hardware button or making it readily available within the UI.

3. For the sake of accessibility, avoid the use of small font sizes in order to cram more information onto the screen. Scrolling in mobile applications can be difficult for the end user, so limit the need to scroll within screens where possible.

Data Access

One commonality between the most modern mobile platforms (iPhone, Android, Windows Phone) is that none of them offer any capability to connect directly to a database – for good reason. The current mobile architecture paradigm simply doesn't support this scenario for modern database platforms in their current state. Given that most mobile applications communicate over the public Internet, access to a database would require exposing that database publicly – and in this age, no sane IT or database administrator would publicly expose an instance of Oracle, SQL Server, or MySQL outside the firewall without measures like a VPN or IP restrictions in place.

Rather than attempting to provide support for database client connectivity, the current paradigm for data access from mobile applications is based on web services.

For the example scenario of extending a common two-tier enterprise application onto a mobile platform, usually, a web services layer would first have to be created that would exist in front of the database or APIs of the enterprise application. In the design of a web services layer for a mobile application, logic around authentication, authorization, validation, and business rules should all be executed on the server-side web services of the extended application. As the web services are now exposed publicly for use by any properly authenticated user of your application, the validity of the data and the user's right to call the web service cannot be trusted without first performing additional server-side checks and can be duplicated on the client side

Security

As previously mentioned, data access on mobile platforms generally requires some form of Internet-facing service or data access point that can be communicated with via a mobile device. Database servers and platforms in their current state are not good candidates for public exposure without additional layers of security that are generally not feasible or cost-effective on mobile devices.

Web servers are generally more hardened to attack and, thus, web services are an excellent candidate for exposure outside the firewall to mobile devices over the Internet.

Another security issue inherent to mobile platforms is the security of data that exists locally on the device itself. Obviously, any mobile device can be compromised much easier than a server residing within a secure data center. If possible, confidential data should not be stored on the mobile device itself and should be stored instead on a back-end server and downloaded to the device when necessary.

If for architectural reasons confidential data must be stored on the device, then measures should be taken to encrypt the data with a key that is not stored on the device, if possible. Fortunately, mobile platform vendors are providing more and more support for automatically encrypted disk storage, which makes implementation of secure data storage on the device much easier.

Connectivity

The final major architecture consideration for mobile applications is connectivity. It can no longer be assumed that the application being built will have access to an "always-on" high-speed Internet connection. In the wild, mobile devices will frequently switch between different types of connections (e.g. Edge, 3G, or WiFi) with wildly varying speeds and will often have no data connection. Often, the implementation of offline access for a mobile application simply doesn't make sense business-wise, architecturally – perhaps the application must have access to only the most relevant and up-to-date data (e.g., traffic conditions), or when data is persisted it must be immediately validated and processed (e.g., stock trades). There are use cases for which offline access is absolutely necessary in order to maintain the end user's productivity. One simple way to design offline access and data synchronization involves the creation of two basic components within the application— a caching mechanism and a queuing mechanism.

Security

As previously mentioned, data access on mobile platforms generally requires some form of Internet-facing service or data access point that can be communicated with via a mobile device. Database servers and platforms in their current state are not good candidates for public exposure without additional layers of security that are generally not feasible or cost effective on mobile devices.

Web servers are generally more hardened to attack and, thus, web services are an excellent candidate for exposure outside the firewall to mobile devices over the Internet.

In summary, while it can be challenging, there are well known solutions for each of the previously mentioned issues. And though each mobile platform will have its own specific best practices for each area, many of the best practices are standard across all mobile platforms, regardless of the technology used.

Tools Available

With the wide growth of mobile and the burst in the number of mobile apps available, it comes as a no surprise that there are plenty of tools available for the creation of these apps. Some are targeted towards the novice, the non-programming person who just wants to try a hand at a simple app and some are advanced and consist of tools to and techniques to make sure the app developer does his job faster but better.

Native tools:

These are the native tools provided by phone OS vendors and are usually the most popular tool of choice for many for reasons of compatibility and future ready. They are usually free and require custom coding of every part of the app. This is best for developers who are experts in their trade and need no other tools to develop their apps.

Although these tools are free for use, the App marketplaces charge the developers annual fees for publishing their apps to the respective marketplaces be it Apple App Store, Google Play Store or the Windows Phone Store. (Blackberry App World submissions being free for now).

Other Tools:

Since the mobile market is highly dictated by iOS, Android and Windows devices, if you want to develop a particular app, you may at least need to develop it for these three devices/platforms unless you can afford to ignore one or all these platforms. Expecting speed and cost-efficiency in native development doesn't make sense since it requires you to obtain respective SDKs and tools of each platform, create multiple code bases and design UI/UX for each platform exclusively.

In these cases, it is more sensible to follow what modern companies and app developers do i.e. 'write once and run anywhere' with cross-platform development since it allows you to develop apps that can run on multiple platforms.

With cross-platform development, you can reduce the cost of development below the threshold of the total sum of native development costs for each platform.

When it comes to cross-platform development for mobile, the most popular frameworks that come to one's mind are Titanium, Xamarin and PhoneGap. All these frameworks solve the purpose of developing a single app for multiple platforms. However, there are vast technical, business and philosophical differences.

PhoneGap

PhoneGap is an open-source and simplest cross-platform framework compared to Xamarin and Titanium. It allows creating mobile apps utilizing Web APIs, i.e. it wraps up web applications in a native app shell and then implements them on native stores for different platforms. It uses a cloud-based service called 'Build' with which you can compile apps for several operating systems without the need to install SDKs of each platform. Any PhoneGap application is simply a collection of HTML pages which is rendered as a Web View. To develop applications in PhoneGap, you need to use HTML5, CSS and JavaScript.

Xamarin

Xamarin, originally called MonoTouch is another cross-platform framework that has picked up the development market with its own IDE. It works on C# within the .NET framework and allows you to create native apps by utilizing native APIs and UIs of each platform. Xamarin comes with Xamarin. Forms library which allows you to write native UIs for once and then share and convert them to platform-specific UIs. Xamarin currently supports iOS, Android and Windows platform. It also allows developing apps for Blackberry by compiling Android apps.

Titanium

Titanium is a JavaScript-based development platform in that, it uses JavaScript to write application codes with native APIs and UI conventions of each platform. This means, it doesn't try to accomplish the notion 'write once and run anywhere' but it attempts to write apps reusing JavaScript with platform-specific features and performance. It is a bit more complicated than Xamarin and PhoneGap there is need to learn the UI API of each platform over and above JavaScript which again is complex for building large apps. Titanium currently supports only Android and iOS.

PhoneGap		Xamarin		Titanium	
✓	Small and simple native API sets enable easy porting to different environments.	✓	Xamarin has TestCloud which allows you to test your apps automatically	✓	Better performance due to native API usage, which also gives access to elements and features of iOS and Android
✓	High reusability with HTML5, CSS and JavaScript. Anything written as a web page can be easily wrapped up as an app.	✓	Provides 100% code reuse with Xamarin.Forms UI development using shared code base and logic.	×	Since it doesn't use HTML5 and CSS, the animations and DOM elements are laggy and less responsive.
✓	Supports all platforms and operating systems which includes iOS, Android, Windows Phone 8, Blackberry, Firefox OS and Ubuntu.	✓	Supports patterns like MVC and MVVM	✓	With Javascript, it ensures quick and easy development
✓	Developers who are accustomed to HTML/CSS/JavaScript, find it easy to start working with PhoneGap.	✓	Xamarin.Android supports Google Glass devices, Android Wear, and Firephone	✓	The look and feel of Titanium apps are better than apps built on other platforms as the UI is essentially native
×	Lower performance of apps as the original codes of the app remains that of a web app.	×	Impacts load time as it has its own runtime	×	Difficulty in developing complex applications
×	Too many fragmented libraries and frameworks at a very basic level	×	Does not provide access to certain Android specific UI controls.	×	No support for third-party libraries
×	User interface of app varies depending on the quality of Web View rendered	×	Does not support sharing of codes outside Xamarin environment for native or HTML5 development		

How much will it Cost?

You've got the idea, you know what you want it to look like, but how much is your app going to cost you to make?

Since Apple launched the App Store in 2008, apps have become more and more integrated into our everyday lives. We use them for almost everything including our work, our leisure time and even our health. Whatever your project be it an event, business, service or promotion, an app can be a seriously potent marketing tool. If managed correctly an app can drive unlimited traffic and revenue your way and its potential should not be underestimated.

While some apps can be phenomenally expensive to develop, others can cost practically nothing. How much should you set aside to create the app you have in mind and engineer an appropriate user experience? To begin with, you should consider the points below.

What do you have in mind

You probably have some idea of what genre or category your app will fall into according to its subject and content, but more important than content is the app's architecture. In other words, the basic framework upon which it will operate.

Although the range of apps available varies infinitely, for the most part, an app will fall into one of the following categories.

- Table-based apps.
- Games
- Database driven apps
- Web Based
- Custom Utilities

Decide whether you want your app to be native to a particular platform. It is worth mentioning that Apple enforces quality control measures on the apps sold in the App Store in a way that is not seen across other platforms. This may or may not work in your favor depending on what sort of app you are creating.

Developer Costs

During the initial stages of development, it is important to consider the types of devices you are targeting with your app, taking into account specific features and constraints and understanding how they work with the scope of your project. Competition means that the market is forever evolving and as phones and tablets change so do their screen sizes. Your app needs to be compatible with all devices past, present and future which carry your chosen platform.

Design Costs

If you are aiming high with your app, good design is an investment. Design can be costly because, with separate screens, icons and buttons apps often have so many elements to design separately. Referring specifically to the cost of designing an iOS app, According to one estimate, figure of between $500 and $10,000 is what we are looking at here, although between 25 and 50 percent should be added on to this figure if you are planning to launch your app on the iPad or other iOS devices. Android apps can cost more owing to the range of devices and sizes the design has to be tailored to.

Total Costs

There are several other figures to take into account. Total costs are difficult to estimate because of the levels of variables involved but outside of development and design, you need only consider the fees charged by your platform (Apple charges 99 dollars per year and around 30 percent of sales) and IT factors like servers and hosting.

How much will it Cost?

As with websites, starting off with an app template can be both cost-effective and a huge time saver. What it means is to basically start off with a clone app similar to your required app. Such templates are readily available at a very low starting cost (some starting as low as $10). If your required app can get a head-start of months just by spending 10—50 $, then why not!

Conclusion:

Apps can be the perfect way to success in guessing by today's technology trends. Utilizing it rightly can amount to a significant potential revenue stream considering it is the platform with largest potential customers. But, having the right approach can be the difference between a success and a failure. With this whitepaper, Pegasus One expects to be a helpful guide for your next big project. Our experts can help you to make an app that is beautiful, functional and state-of-the-art.

Pegasus One

1440 N Harbor Blvd #900, Fullerton, CA 92835, USA

Phone: +1 (714) 485-8104

Email: info@pegasusone.com