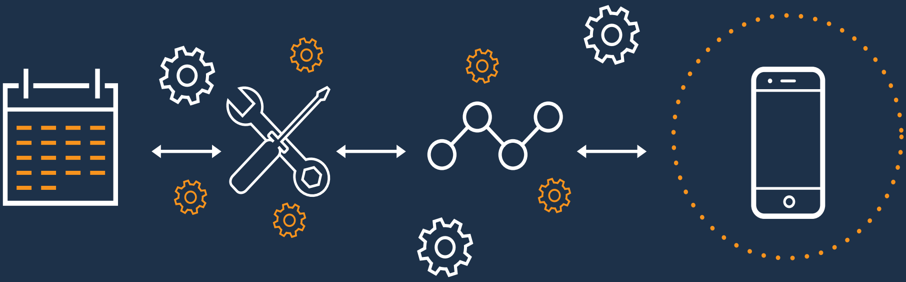
VERSION
2.0

Segue Technologies, Inc.

PLANNING YOUR MOBILE APPLICATION



www.seguetech.com
Version 2.0

Published 2016 by Segue Technologies, Inc.

Cover Design, Layout, and Illustrations created by
Segue Technologies.

All links featured in this book can be found at
www.seguetech.com/blog

About Segue Technologies
Segue has developed innovative, dependable, and user-friendly applications since our founding in 1997. We provide a wide range of Information Technology services, focusing on Software Engineering, Information Management, Quality Assurance, and Systems Integration. We are a growing small-business, supporting Federal, Commercial, and Non-Profit clients.
www.seguetech.com

TABLE OF CONTENTS

MOBILE APP PLANNING

5 Things You Must Know Before You Start

1

Why are you building the app?

You should have a clear vision for the app and the business goals it will help to achieve.

2

What is the target functionality of your app?

Focus on the functionality that will deliver the biggest impact.

3

Should you create a Native App or a Mobile Web App?

Having a clear understanding of your target audience will ultimately guide your platform decision choice.

4

Does your App Developer have a good understanding of Design & Usability?

Your app developer should have an understanding of design options and constraints to create the best experience possible.

5

Do you need a Privacy Policy?

Collecting sensitive information? You must include a privacy policy that explains what information is being collected and how it will be used.

Illustration by Segue Technologies

FOREWORD

It’s a good time to be in the Mobile Application Development business, as the world has definitively shifted to a mobile-first mindset. New ideas arise every day for both new app concepts and transforming old tools into mobile apps. Segue is regularly introduced to passionate people with app ideas that could revolutionize aspects of our daily lives and solve any of the numerous problems that exist in our society. Appreneurs come to us to develop their ideas, which may seem fairly simple, but can be much more complex from a technical solution standpoint.

In the recording industry bands are told that studios are too expensive to use as a rehearsal space. The same goes for Mobile App Development; knowing your material before you come in can save you a fortune. Bringing app-based businesses to life is not usually a “cheapest option” activity and trying to work out the details during development is likely to be very costly, and can even be disastrous. To prepare, your development team should use a Discovery process to fully understand your business’ needs and the associated app requirements. This can be perceived as added cost to the development, but it truly ensures the best product is created and it saves greater expenses related to redevelopment, or worse a final product that fails. Think about it. We’re talking about your business-your big idea- so it’s necessary for a company to go beyond the bare minimum functionality to help you create something polished, user-friendly, and competitive.

This eBook was developed to help you better understand the process of developing a custom native mobile application. It applies to people in all stages of concept development including: the initial idea, accurately defining your requirements, choosing a development company, technical development, quality control testing, marketing, and your app launch. Having a better understanding of the steps involved and the high-level processes can ensure a successful project aand help you stay within your budget.

Ron Novak
Executive Vice President

Part I

Planning and Preparation

When to Outsource Your Mobile Application Development

While many companies are able to sustain internal development teams to support their business systems, they don't always have the capacity or specific experience for every project. Before you task your team with a new project, such as a custom mobile application, you may want to consider outsourcing.

What is Outsourcing?

According to Entrepreneur.com, Outsourcing is defined as "the practice of having certain job functions done outside a company instead of having an in-house department or employee handle them; functions can be outsourced to either a company or an individual." This is a way of augmenting an existing technical team and/or lowering costs. Also, depending on the nature of the project, outsourcing might be a viable option to ensure rapid and successful delivery.

Outsourcing Considerations

There are a number of factors to consider regarding outsourcing mobile development:

- Availability of required technical resources and the project timeline: This is a fairly obvious consideration. If in-house resources are participating in other projects, you may need to add other developers to meet workload and timeline requirements.

- Customer's understanding of requirements: Most customers have a basic idea or goal for an application, but do not always understand the detailed requirements. This can affect the feasibility of using outsourced resources. In one scenario, an outsourced team may have deeper experience in the type of project and is, therefore, better able to understand, define, and execute on the project requirements. On the other hand, if you do not understand the full scope or need for your project, you may not be ready to commit funds to an external team, as you risk costly changes and unforeseen issues arising.

- Expected stability of requirements: Related to understanding your project requirements is an assessment of how stable your project requirements are likely to be. For apps that are market-driven, requirements can change frequently over the course of a development project. In this case, using a non-co-located resource may not be recommended, due to the need to respond quickly to changes, and potential communication challenges or delays. However, you may still want to consider going to an external resource to augment your in house team with capacity and specialized capability.

- Functional or technical complexity: Sometimes app logic is very complicated, or app development requires access to a specific operational data store or customer environment. This can preclude the use of external resources for logistical or security reasons.

- Cost: This is the driving factor for most outsourced development decisions, and it is a compelling one. In many cases, outsourcing development can cut costs because you may only need the resources on a short term basis and you eliminate the overhead costs that may be associated with hiring full time personnel yourself. They also may be more affordable due to the internal efficiencies of the outsourced development team, or due to a lower cost of living in their specific locality. However, although developer costs can be lower, we generally expect that additional time and therefore cost) will be incurred for project management, requirements management, and quality assurance during the project. So, it is important to also assess price reasonableness when getting a quote to ensure that you actually get the product and quality you need.

Segue evaluates each app development project and works with our customer to determine the most balanced approached to completing the work. Cost is important, but our primary focus is on delivering quality apps that will satisfy customers' needs.

Mobile App Planning: 5 Things You Must Know Before You Start

In all likelihood, your app development project falls into one of three categories. It is either a consumer app that has the potential for widespread adoption; a modernization effort of a new or existing business process; or the migration of a web application to a mobile-optimized version. For the sake of this article, I will assume that you already have a budget in mind (we will lay out the factors that affect cost later in this eBook). So now what?

You need a sound plan to turn your app idea into a reality. All too often, organizations rush their mobile app to market without considering several key factors. Lack of planning is the easiest way to mismanage an app development project and will ultimately create re-work, diminish profitability, and decrease utility. It is critical, therefore, for your organization to have a mobile application development plan in place that addresses the following questions:

1. Why are you building the app? This might seem like the easiest question to answer. However, I can't tell you how often we have customers tell us that they need to build a mobile app, but can't articulate its specific purpose or how it is different from what they offer on their website or what is already available through their existing business system(s). It's important to have a clear vision for the app, as well as the business goals it will help to achieve.

2. What is the target functionality of your app? Does the functionality proposed provide value to your organization

or customers? Will people want to use it? If the functionality proposed doesn't increase efficiencies, make routine tasks easier to accomplish, or provide some other value, it is unlikely to be used. Focus on the functionality that will deliver the biggest impact instead of adding needless features that are unlikely to be used.

3. Should you create a Native App or a Mobile Web App? If your app needs access to device-specific hardware, such as cameras, barcode scanners, etc., you might be better served to create native apps for your desired platforms. If you are looking to simply provide an optimized mobile experience for your existing website or light-weight functionality, a mobile web app might be the answer. If you decide to go Native, do you plan to build for iOS and Android or do you also want to port to Windows Mobile? If your target audience is medical staff that exclusively uses hospital-issued iPads, it doesn't make any sense to develop your app for Android. If your app is projected to have widespread adoption and you have the appropriate budget, building separate instances for each operating system might be necessary. Having a clear understanding of your target audience, to include usage patterns, user scenarios, and device use, will ultimately guide your platform decision choice.

4. Does your App Developer have a good understanding of Design and Usability? If a mobile app is not aesthetically pleasing, intuitive, and easy to use, the ultimate success of the app will suffer. It is critical that your app developer has

an understanding of design options and constraints of the targeted platform to create the best experience possible. It is also very important that usability is taken into consideration early on in the project. If you build your app correctly, the user should be able to perform the intended functionality without assistance. Ensuring that the application was developed to optimize screen real estate, use appropriately sized buttons and fonts, and use intuitive navigation between tasks will help create a user experience that users will enjoy.

5. Do you need a Privacy Policy? If you plan to have your app collect any sensitive information from users, you must include a privacy policy that explains what information is being collected and how it will be used. Delta Airlines found this out the hard way when the State of California sued them for not including a privacy policy in their “Fly Delta” app.

Part II

Time, Cost, and Complexity

How Long Does it Take to Build a Mobile Application?

While there are a variety of factors involved in building a mobile app, in our experience at Segue, it takes at least three months to build the initial version of a mobile app, and up to six months or longer for apps with more features and advanced functionality. To understand how long it takes to build a mobile app, let's take a more detailed look at the production process and other project factors from start to finish

The Production of Mobile Apps

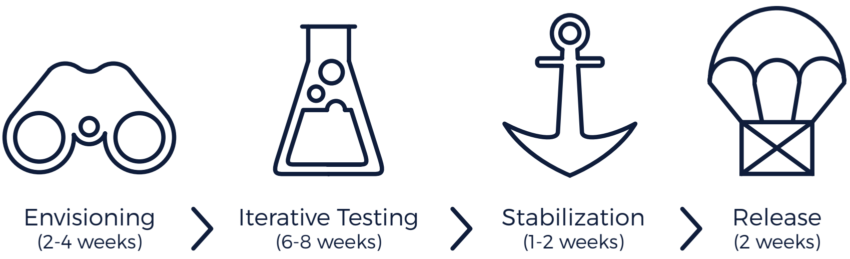
Segue uses an agile software development approach when developing mobile apps, which has a number of benefits, including being able to work closely with our clients throughout the duration of a project, providing incremental value and demos, and the ability to react to changes along the way. However, while using an agile software development approach streamlines the time it takes to build a mobile app, there are still a number of steps involved which add up, including:

Envisioning (2-4 Weeks): Defining a clear vision of the client's objectives for the solution is one of the most critical stages in the lifecycle of any project. Through a series of discovery sessions with the client, the project team ensures a comprehensive understanding of the client's vision, requirements, and design, required for a successful solution. During the envisioning phase, requirements are created in collaboration with the client, creating business-focused acceptance criteria for each requirement. Additionally, this phase includes creating design concepts and wireframes for the website's look and feel, navigation, layout, and technical architecture.

Iterative Development and Testing (6-8 Weeks): After Discovery, the project team begins development and testing of the mobile app. The multi-disciplinary project team consists of analysts, designers, developers, and testers that work in time-boxed iterations to implement a subset of the overall requirements, concluding with a client review and demo to elicit feedback and plan for the next iteration.

Stabilization (1-2 Weeks): The final development and testing iteration is focused on stabilizing the mobile app, making final changes based on client feedback and ensuring compatibility with mobile devices.

Release (2 Weeks): After development and testing is complete, the project team assists the client to release the mobile app to the Apple App Store, Google Play Store, or other mobile app marketplaces. The Apple App Store may take up to two weeks before it approves your application for public download.



Mobile App Features, Complexity, and Number of Users

In addition to the production process itself, there are other factors that determine the time it takes to build a mobile app, including:

Features
One of the primary determinants of how long it takes to build a mobile app is the number of features involved. A feature is functionality in the app, such as the ability to buy an item, send a Tweet to Twitter, or scan a barcode to look up a product's price. As you can imagine, the more features an app has, the more time

it takes to design, develop, and test. We encourage clients to focus on building Minimal Viable Products - the minimum amount of functionality required to provide value to your customers. Building an MVP keeps the overall number of features focused, creates the first version of your mobile app sooner, and allows additional features to be developed through subsequent releases as you learn more about your user's needs and feedback.

Complexity

In addition to the number of features, the complexity of each feature also determines the time it takes to build a mobile app. For example, a feature that allows a user to look up the price of a product using a text-based search is likely less complex then looking up the same product using a photo. The latter may provide a better user experience, but might take longer to build.

Number of Users

In addition to features and complexity, the number of users also impacts the time it takes to build a mobile app. Building an app that can handle 500 simultaneous users has fewer factors involved than building an app that can handle 5,000 or 50,000 simultaneous users. Additional investments in architecture, performance testing and tuning, and infrastructure are required to support a large number of users – all of which take more time.

iOS, Android, Web – Or All Three

To reach the widest possible audience, our clients often request Android application development, iOS application development and sometimes even web-based apps as well. While there is some efficiency to building multiple apps at the same time, the design, development, and testing activities are unique to each platform, requiring separate effort. While some of these activities can occur in parallel, extra time is often required to build two or three apps.

Resource Availability

When producing a mobile app, Segue prefers to have our team members dedicated to the project, ensuring they can closely collaborate with our clients and other team members. However, our team member availability varies based on active and planned projects. The case may be the same for other mobile development companies and, therefore, there may be a delay in starting a mobile app project until team members are available and can give their complete attention to the task at hand, adding to the time it takes to build a mobile app.

Client Availability

We believe in close collaboration with our clients and their users. However, this requires a commitment of time and availability from our clients. By being available for a few hours a week for a few ad hoc questions, the time it takes to build a mobile app can often be decreased, reducing delays caused by waiting for feedback and clarification.

While we are always excited to release a new mobile app to the public, as you can see there are a variety of factors involved in determining the time it takes to make a mobile app

How Complex is my Mobile Application?

One of the biggest variables in the cost and time required to create a native mobile application is the complexity of the application being developed. Just like any other type of software development, simple mobile applications can be completed quickly while complex mobile application development can take a significant amount of time to finish. With all the tools, libraries, and advanced development environments available to a developer, it's hard for the uninitiated to tell what's complex and what isn't. By the end of this article, you'll no longer be uninformed and you'll be able to tell what makes a mobile application complex.

Context Entities Features

Context

First, you need a context. You could keep reading without one, but to really appreciate how to evaluate the complexity of a mobile application, it will be better if you have a specific mobile application in mind. If you have a mobile application idea that you would like to see in the market (or bring to the market yourself), that's perfect. If not, just pick your favorite mobile application and follow along.

There are two main areas of concern when determining the complexity of a mobile application: Entities and Features. The entities of a mobile application include the data and changing information that make the application interesting – these are the items in your online shopping cart or the tweets of those you're following. The features of a mobile application, on the other hand, include what it does and how it does it. Separating these out is normally part of the analyst's job: breaking the application down into actionable components. We're going to walk through a lightweight way to break these down so you can get a peek at what makes a mobile application more or less complex.

Entities

In most applications, the user is interacting with different collections of information for different purposes. A weather forecast is a collection of information like high and low temperatures or whether it will be sunny or rainy that day. A location can be a collection of information like city, state and zip code, or even latitude and longitude. A person might be represented by a name, title, and phone number. Each of these collections is combined into something called an entity, and each application may track different types of entities. Defining entities can get really messy, even for programmers, but for the purposes of this blog, just keep things simple and define entities as they make sense to you.

Also, we're really only concerned about entities that can change or entities with values that can change. For example, if you're dealing with an application that displays address information about the

White House on its own dedicated page, that display does not use an entity. If instead the application shows the address information for different popular DC landmarks depending on which one is selected, then the display does use an entity.

It's fairly intuitive to see that the more entity types an application has, the more complex it is. What is not so obvious is that this increase is not a one-to-one relationship. Normally, things get increasingly complex as more entity types are added. For example, an application with two entity types might be twice as complex as an application with one entity type, but one with three entity types might end up being four or five times as complex!

Features

Most of the features of a mobile application are designed for the target user: what should the user be able to do with the application? Some of the features are oriented towards the owner (or creator) of the application instead of the end user. For the sake of brevity, both are included in the same list. The list is ordered from least complex (XS) to most complex (XL). Take a walk through the list and see how complex the different features of your application are to design, implement, and test. Where applicable, we'll use a basic weather forecast application as context for an example.

EXTRA SMALL

Feature	Description	Example
Call a phone number	If it's a smartphone, it's easy to call a phone number stored in the application so the user can talk to the person on the other end.	Including the phone number of a tornado watch service can make it easy for users to report tornado sightings.
Start an email	If the user has an email account on the mobile device it is pretty straightforward to start a new email for the user to send.	Including a feedback email address makes it easy for users to provide feedback on the quality of the app.
Open a webpage	If the mobile device has a browser, then opening a web page is a snap. This is often done to take the user to a page with more verbose and current information on a topic.	If a weather application only shows summary information, then the user could be presented with an option to view the full forecast on a website in the browser

SMALL

Feature	Description	Example
Preferences	This is the little configuration or settings pane that enables users to store details on how they want to use the application	Users can choose to view their forecasts using either Fahrenheit or Celsius.
List of entities	Once the entities are in the application, it's easy to show them in a list.	Users can keep a list of cities to easily see forecasts for different locations quickly.
Re-sort entities in a list	Re-sorting simple entities is straightforward, but complex or interdependent entities can make re-sorting a nightmare.	Locations are sorted alphabetically by default, but the user can re-sort them by distance from the device.

Driving directions	Most devices include GPS, built-in maps, and navigation. Applications can take advantage of this to provide the user with directions to a specific location.	The user can quickly get driving directions to any city chosen as a location for forecast information.
Basic search	When the user is presented with a list of details or options, a search field allows the user to enter text to filter the list to the items that they are interested in.	When presented with a list of locations with forecasts, the user can start typing the name of the location instead of having to scroll through the entire list to find their target location.
Attaching a gesture to an action	Lots of people love the different gestures that are available – pinch, long press, swipe, shake - and it's fairly easy to attach them to an action in a mobile application. While these gestures should only be included as part of a well-planned user experience, they can also make the application more fun and intuitive to use.	Shaking the device could trigger an update to get a more current forecast
Facebook and Twitter	Letting users post status updates and photos to Facebook or Twitter from within the application can make the application feel more familiar to users.	Once the user has granted the application access to their Facebook or Twitter account, they can post their feelings about the current weather conditions from within the application.

MEDIUM

Feature	Description	Example
Map	Showing nearby entities on a map is a standard mobile application pattern that can generate a high “wow” factor for the amount of work involved. Showing more than a dozen entities will make this more complex.	The locations a user has picked for forecasts can be shown on a map with icons indicating the current conditions. When the user selects a location, they are taken to the full forecast for that location.
Advanced search	This is a basic search with the added option to filter lists based on selected criteria.	When presented with a list of locations with forecasts, the user can select a US state to filter the values displayed, and then start typing a location name in a search field to find their target location.
Basic analytics support	This one is for the owners of the application and it enables them to see how the application is being used, what the most popular screens are, and where people are using it from. Analytics can give insight into how popular different parts of the application are and uncover any problems users might be experiencing.	With analytics, the owner of the application can tell what the most popular times are for checking the weather and use that information to make other decisions such as when to schedule server-side updates.
Display entity details	If the entity only has a few fields, then showing that information is straightforward.	Users can view the details for a forecast such as temperature and chance of rain.
Photo slideshow	Showing photos requires different interfaces than normal text-based data. Users typically pick a photo from a grid and swipe to look at the next photo in the list.	Users can see the photos of interesting weather for each of the locations they are tracking.

LARGE

Feature	Description	Example
Tablet support	Providing a customized experience for tablet users makes it more appealing to those with tablets.	When used on a tablet, the weather application has different navigation options, uses much larger images, and provides more details about the weather conditions.
Notifications	Push notifications send versatile application-centric information to the user in a timely fashion. The server infrastructure support required to facilitate sending the notifications makes this feature somewhat complex.	Users can choose to receive alerts about severe weather in their selected locations.
Sync with multiple devices	There are a variety of methods to sync information across different iOS or different Android devices. However, syncing between the two platforms is a much more complex undertaking.	If the user has multiple devices, they can sync the locations they are tracking across all devices.
In-app purchases	The ultimate monetization method: letting users buy things from within the application! Use this to connect your users to a variety of products and services that they are willing to pay money for.	Users can purchase prints or high-resolution images of various weather-related pictures from within the application.
In-app ads	Giving some of the screen space of your application over to advertisers can be a lucrative way to monetize your application.	When it's sunny, users may see an advertisement for sunscreen on the lower portion of the screen.

Account Management	Some functionality requires each user to have their own account to manage their own version of the application. This is important for applications that can be accessed from other devices or web browsers as well as applications that store and manage a lot of information that is personal to the user.	If the user wants to upload their own weather-related photos, they need an account to manage their uploaded photos and control access.
Offline data support	Most applications take the easy way out and only show data (or entities) when it can directly connect to the internet. If you want users to have access to data when they are offline, some extra work has to be done which increases the complexity.	A user can still access all their uploaded weather-related photos even when they are offline.
Create, edit, and remove entities	Allowing users to manipulate data makes the application more complex than some people realize. Providing appropriate entry fields, validating data that is entered, and properly handling removed entities all require significantly more effort than just displaying read-only information.	Users can record and manage additional information about their uploaded weather-related photos like image title, description, date taken, and camera used.

Hopefully, now you have some idea of what can make a mobile application complex. Obviously, this list doesn't cover everything that a mobile application can do and it certainly doesn't replace the need to talk to an experienced mobile application provider (like us) to get an estimate on complexity, time, and cost for your next mobile project.

How Much Does it Cost to Build a Mobile App?

Have you noticed how very few companies in the software industry address the question of mobile application development cost on their websites? If you have searched on the internet or spoken with a software development company directly, the answer you probably have encountered most frequently is; “well, that depends” with little to no additional explanation. While this may seem like a cop out, it’s actually the right answer- it does depend. Don’t stop reading; I promise I’ll explain further. Now, I might be breaking some unknown pact between my fellow software brethren by revealing this super-secret information, but the majority of the mobile application development projects that we have worked on have fallen within the range of $100,000-$250,000. This is primarily because we focus on enterprise mobile applications and many of our partner companies in the industry develop less complex apps for much less.

Now let’s get into what exactly “it depends” on. The three main factors that affect the cost of a mobile app build-out are: Functionality, the targeted users/devices, and timeline.

1. **Functionality (what you want the app to do)** is the most important factor in determining cost. There is a big difference between an app that does something simple (such as creating a to-do list) compared to a fully-featured interactive app like the new Star Wars app. Historically, apps that perform one or two functions really well (such as Uber or Words with Friends) have been more successful than apps that have been crammed with extra and sometimes needless functionality. Although app developers are getting more efficient with usability when delivering apps with more extensive functionality, I still think that developing an app to do one or two specific tasks really well is the best approach and ultimately will reduce the overall cost of development. The main take-away here is the more you want the app to do, the more it is going to cost you.

2. **Targeted users/devices (who you want to use your app)** can drastically affect cost when developing a mobile app. There are a few key technology-based questions you should be prepared to consider when planning your mobile project. The biggest and most commonly debated question is whether you should build a Native App or a Mobile Web App. For reference, a Native App is one that is developed specifically for a certain mobile operating system (such as iOS or Android) and installed directly on the device whereas a Mobile Web App is an app that is not operating system-specific and runs on a device's web browser. Ultimately, cost will be affected by the selection of the target audience since this selection drives the technical solution. For example, let's say you want to develop a clinical mobile app to assist Doctors with conducting their rounds and you know that they will all be using hospital-issued iPhones. In this scenario, developing a Native app for iOS (Apple) devices seems like a pretty logical solution, right? However, take that same scenario and complicate it by requiring support for Android, Windows Mobile, and

Blackberry devices. If you were to develop a Native app for each of the required platforms it could end up costing you 4 times what it would cost to develop one mobile web app! The reason for this is that code development for each mobile operating system is essentially an independent effort and usually requires different programming languages and environments. So, as you can see, one size does not always fit all with respect to mobile and the selection of your target audience weighs heavy on cost.

3. Timeline (when do you want it!) is another key factor in cost determination with a mobile app project. Having a reasonable timeline (with customer involvement throughout the process) is the best way to keep the project within budget and avoid cost overruns. Having an unrealistic expectation of a project delivery can cause unneeded rush charges and can also affect cost. For example, let's say your company wants to develop an app to control the Mars Rover and you want it next week. Do you think a rush delivery affects cost? You bet it does. Although some development tasks simply can't be rushed and take however long they are going to take, requesting a quick turnaround time can definitely have an impact on cost.

So there you have it folks, a break-down of what goes into the pricing and cost factors of developing a mobile app.

Part III

Decisions to Make

Native vs. HTML5 Applications: Which Approach is Best?

When we're approached about building a mobile application, the first question we often get is, "How much will it cost?" As we discussed in the last section, one of the factors that dictate cost comes down to what type of mobile application you want built.

There are essentially three different types of mobile applications:

- **Native applications**
- **HTML5 applications**
- **Hybrid applications**

Native Applications: Benefits and Drawbacks

A native application is installed directly onto your phone. They're usually designed and built specifically for your device. They're also not cross-platform, meaning you can't build an iOS app and have it run on an Android device - a separate Android application is also needed.

Native applications can make better use of your device's hardware capabilities (GPS, microphones, accelerometers, etc.). They're best suited for rich user experiences and can provide a level of interaction HTML5 apps cannot currently reach. Native applications offer the developers and users the ability to use standardized UI controls, making interfaces more natural to users. These types of intuitive interfaces typically make users very happy.

Benefits:

- Available through standard device marketplaces (iTunes store, Google Play store) which allow for paid applications or licensing fees that can supply the developer with a revenue stream
- Allow for a richer user experience
- User interface is designed for a single targeted operating system, allowing for familiar UI controls to be used
- Can support offline usage (usable when there is no network connectivity)
- Provide increased security options (data at rest and in transit can be encrypted)
- Background processes can be leveraged to provide push notifications, monitor for specific events

Drawbacks:

- Require separate development efforts for each supported operating system
- Market fragmentation could limit your target audience if you're trying to use newer device/OS features
- Application updates must be pushed for each platform through the app stores
- Increased costs to support multiple platforms
- Applications must go through app-store approval processes (iTunes, Google Play)

HTML5 Applications: Benefits and Drawbacks

These applications are written in HTML, JavaScript, and CSS. They run in your device's web browser, not within their own container, and allow developers to create a single application that can run on almost any device. This keeps developers from having to develop, maintain, and publish separate code bases for each operating system.

However, with an HTML5 application, there are a number of tradeoffs that must be made. For example, iPhone users like their tabs at the bottom of the screen. Android puts them at the top. With a single application, designers have to make decisions that could leave one group feeling left out. Trying to blend the two interfaces also leads to a seemingly muddled interface. Usually, the best approach here is to be generic enough that the app doesn't feel too much like iOS or Android. Customized graphics, navigation controls, and icons can make your app feel more natural to users. Those things, however, require additional development, since you're not able to leverage the native UI controls.

Benefits:

- Single application for all users, regardless of operating system, hardware, or network.
- Faster time to market.
- Reduced development costs.
- Centeralized code base does not require “updates” to be pushed to each user. They’re always on the current version.
- No app store approval process.

Drawbacks:

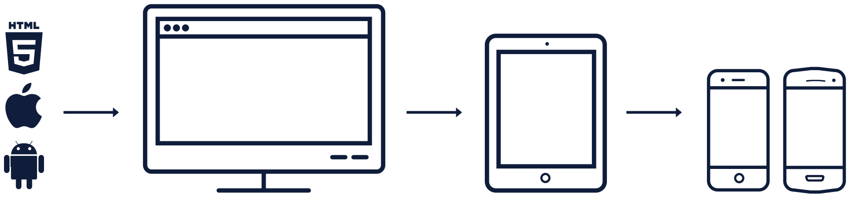
- No centralized “App Store” for HTML5 applications, resulting in reduced monetization options and no built-in support for in-app purchases - you must process all transactions externally.
- Can’t leverage app store for marketgin options/search availability.
- Device browser inconsistencies don’t truly allow for a “write once, run everywhere” approach. Testing and optimization for each browser is still required.
- No background processes are available. Users must have your app open in order for you to interact with them. This includes push notifications.
- User interface must be consistent for all platforms, usually causing one group to feel alienated.

Hybrid Applications: Benefits and Drawbacks

A hybrid application is essentially an HTML5 application inside a native application wrapper. This approach allows developers to leverage much of the benefits of a native application (hardware availability, background processes, etc.) while still using HTML5 for the majority of the content.

Hybrid apps can be a bit tricky sometimes. Apple's approval process might not let your app into the iTunes store if they don't think you're offering enough value to your users. They might tell you to just launch it as a web page if your application only provides an icon and a web view to your HTML5 site. Android is a bit more lenient on this, but users are typically not as receptive. Mobile users have been getting much better at identifying those apps that only provide a bookmark to a mobile website.

The best time to choose a hybrid application is when your content is more dynamic and will change constantly, but you want your users to have a richer experience than what they'd normally get on a mobile-optimized website. Allowing them to store login credentials, maintain notification settings, access GPS data, etc. are all examples of value that can be added to a basic HTML5 application.



Benefits:

- Shorter development time than a fully native application.
- Interface could use native controls, or be built fully in HTML5. These are the major design considerations.
- Support for more dynamic content.
- Background processes and notifications can be leveraged.

Drawbacks:

- Developers still need to maintain multiple (albeit smaller) sets of code for each platform.
- Applications must go through approval processes for app stores.
- User experience may not be as rich as many users expect from a native application.

In conclusion, the type of application you develop really depends on what you want your users to do, your time frame, and your budget. Native applications can provide the most rewarding experiences, both in revenue streams for you and the experience for your users, but can be costly. HTML5 applications can provide the "good enough" experience, but may leave users feeling like they're missing out on something. Hybrid apps can provide the middle ground between the two, but must be very carefully designed in order to create a successful user experience in all platforms.

Which Platform Should You Develop for First: Android or iOS?

One of the questions we at Segue are often asked by our customers is, “Which platform should I build for first, Android or iOS?” There are many different aspects to consider when answering this question. To answer it fully, we’ve brought together two of our mobile application developers: Geoff Bender, our lead iOS developer, and Mark R. Andrachek, Jr., our lead Android Developer.

Round 1: Market Share

Just how many users are there in each ecosystem and how many devices are there in existence? How has this been changing over the last few years? Where do you see this going in the future?

Geoff Bender (Lead iOS Developer): People often ask who is winning the market share battle but it’s really difficult to answer that question. In the PC world, market share used to translate directly into financial success but in the mobile market it appears to be the opposite. Android appears to be winning the market share battle at this point but iOS users appear to be spending the most money on apps. This may be due to Android market share being stronger in relatively lower income countries or possibly because Apple consumers tend to spend more money on devices, leading to more money being spent on apps. If you plan to profit from your app, iOS would probably have the biggest impact on your bottom line.

If you plan to distribute your app for free and wanted to reach the widest possible audience, especially outside of the US, than perhaps Android is the way to go. They're really dominating the world market at this point. There have been some studies that speculate that customer loyalty to Apple is going to win out over time and Apple will command the majority market share within the next several years, especially as they aggressively move into China, but it's almost impossible to predict that sort of future.

***Mark R. Andrachek, Jr.* (Lead Android Developer):** Android currently dominates the global smartphone market, with about 80% of the market and well over 1 billion users. This makes Android the most popular operating system on the planet. Apple likes to toot their own horn about their high adoption rate for new OS releases - but to put that in perspective, the number of users just on Lollipop (Android 5.x) eclipses the total number of iOS users.

Depending on who is doing the measuring, Android has eclipsed iOS in web traffic as well as advertising-based revenue. However it still lags slightly behind iOS in terms of app sales revenue, with iOS users being more likely to spend money on apps (unless you include third party app stores).

Round 2: Software and Hardware Options

What are the different software and hardware features that iOS and Android provide, respectively?

Andrachek, Jr: Because of Android's popularity there are many Android devices out there — phones, tablets, and wearables —

in just about every size and price point imaginable. This is possible due to the open nature of Android. It is built on Open Source Software, including the Linux kernel, and is itself largely available under the Apache 2 license.That open source nature has made Android the de-facto choice for companies and users looking for customizability and flexibility. You have third party app stores, like Amazon, F-droid, and the region specific markets like those in China. There are custom ROMs that replace the entire OS, third party launchers (home screen replacements), home screen widgets, theme and icon packs, and hardware with customized versions of Android like Amazon's Kindle Fire, NextBit's Robin, and SilentCircle's Blackphone.

While this diversity can be a challenge from a support and testing perspective, it's also one of the core strengths of Android. Competition between Android device manufacturers, not just iOS, has helped drive progress. Important bits, like fingerprint support for example — once requiring one-off manufacturer specific SDK's — wind up baked into the core of the platform.

Bender: Well, many of the devices on the iOS side are similar. Apple controls both the software and the hardware so you don't have quite the level of diversity between the devices and the features that each support. Since the iPhone 5, screen sizes of iOS phones and tablets have varied considerably. I think the bigger screen sizes on Android ended up being a pretty successful feature, so Apple decided to follow suit with that.

Unlike Android devices, Apple had not supported near field communication for mobile payments until the iPhone 6 came around.

Now iPhone users can make mobile payments using any phone newer than the iPhone 6 or by using an iPhone 5, 5C, or 5S in combination with the Apple Watch, although the latter case does not support Touch ID.

I'd say the biggest difference between the two platforms is ultimately the ecosystem and how users organize their lives around their devices. Apple users tend to have iPods, iPhones, iPads, Macs… and all of the devices synch across iCloud. Charging adapters and earbuds can be shared between devices, software usability across devices is consistent, and as time progresses, feature parity between iOS and OS X has grown steadily. Because the hardware and software is created and controlled by Apple across each device, the experience is consistent across the entire ecosystem. Android users tend to use a whole suite of different services, mostly Google services, for organizing their lives but can potentially run into feature differences between various hardware manufacturers and OS versions.

Round 3: Cost and Time

Is there a difference in cost and time when developing for each of the two platforms?

Bender: Development time should be roughly equal. I guess it all depends on the team and how well they know their platform. As far as distribution goes, it all depends on whether you want to submit it to the Apple App Store, or if you plan to distribute it internally and develop software within your own organization- sort of like an enterprise distribution model. They also have a business-to-business distribution model that puts apps on a private App Store, independent of the public App Store.

As far as time goes, Apple requires a code review before posting the App on the store. This could take anywhere from one to two weeks, roughly, depending on a holiday schedule. App updates have to go through the same review process, but that usually happens a little bit faster. Another cost factor is if you plan to sell your app, or content within your app. Apple is going to take a 30% cut of that. Google does the same thing as far as I know.

Andrachek, Jr: Android, with its open source nature and Java base has also resulted in a plethora of libraries that help developers rapidly implement features, in some cases shortening the amount of development time in comparison to iOS (although this tends to wind up fairly even when the greater testing and support burdens are factored in).

It's also important to note some major differences in app distribution. While the Apple App Store and Google Play are both similar in terms of the cut of sales they each take, Apple is far more restrictive in what's permitted in the store. Google is much more open, and they even make it fairly easy to distribute applications outside of the app store allowing for direct sales of apps and content that they do not permit in the app store, as well as making private in-house distribution less burdensome.

Round Four: The Final Decision

In your expert opinion, should you develop for iOS or Android first? Is it a two-for-one special?

Andrachek, Jr: If you want your application to have the broadest reach it makes sense to develop for Android first. If your application is ad-supported, it makes sense to develop for Android first. If your app is not location specific, and you want to have a global reach, it makes sense to develop for Android first. If you have specific hardware you want to leverage that's only available on Android, it makes sense to develop for Android first.

Lastly, you need to evaluate your app's competitive position - it may be that Android is underserved in comparison to iOS (or vice-versa) - and this evaluation may override any other considerations

Bender: I'd say in a lot of ways it is a two for one special. I mean as far as project management and requirements analysis- those can be shared. The graphics team can also be shared. The overall general design is going to be very similar, but each platform has its own set of interface guidelines so there will be some stylistic differences. If you have a backend database with an API, both platforms should be able to utilize that without much trouble. A lot of the logic and the algorithms- how you approach solutions to problems- can be shared between the two platforms. The testing department should be able to test both platforms equally as well, with differences really coming into play if automated and/or unit testing scripts need to be written. Really, the biggest difference between iOS and Android just comes down to language syntax and user interface experience- how they (users) interact with the device.

How Many Versions of iOS Should My App Support?

As developers, we're always looking to take advantage of the latest technologies. Each new version of Apple's mobile operating system (iOS) introduces new APIs that give us the ability to do wonderfully new things within our apps, but the balance lies in leveraging this new technology while still supporting an established user base. We often try to guess how many versions of iOS our apps need to target because the overall goal is to introduce new functionality while supporting the maximum number of users. Adding bells and whistles won't do us much good if that means cutting out a significant portion of our audience. Personally, when trying to decide which iOS version my apps will target, the three primary things I take into account are a high adoption rate, app complexity, and selective feature offerings.

High Adoption Rate

Apple's devices aren't "fragmented" in the way that many competitors' devices are because Apple controls both the operating system and the hardware it runs on. As a result, there are really only two things that keep iOS users from upgrading: either the hardware is sufficiently old to prevent the upgrade, or users themselves choose not to. This clear upgrade path results in a new operating system adoption rate that is staggeringly high. Several sources (such as CNET and ZDNet) claim that 2015's release of iOS 9 reached over 60% adoption within a month's time, significantly faster than that of iOS 8. This works in our favor because we can rest assured that by the time we've developed and tested features introduced by the latest software release, a good portion of our target audience will be able to take advantage of the latest features.

App Complexity

Will my app even need to take advantage of the latest technologies? What if I just plan to present a web view and a few buttons and labels? What if my app consists primarily of table views and doesn't even rely on network connectivity? If I'm following the KISS principle (Keep it simple, stupid!) and I don't plan on doing anything cutting edge, then I could honestly support several prior versions of iOS since most of the basic features have been around since the beginning.

Selective Feature Offerings

If I need to add some newer capabilities, I can always choose to code them in for newer devices and filter them out for older ones. Xcode, the iOS development tool, automates so many of the little tasks that we would have to perform when trying to support multiple devices or versions of an operating system. Want to use the latest software development kit (SDK), but support earlier operating systems? Just set the deployment target to an earlier version with one setting and test for compatibility in the various simulators. It really is quite easy to offer new features to those devices that can support them and selectively filter out features for devices running older versions of the operating system. Many times, the compiler will alert us if we try to compile a feature that isn't compatible with the specified target. Want to take advantage of modern alert views with inline callbacks but you're worried that older devices won’t support them? Just test if NSClassFromString(“UIAlertController”) == nil to determine what the device can and can’t support. This way we can still use the latest features and simultaneously support earlier operating systems.

Potential Pitfalls

Abrupt Transitions

It won’t happen often, but sometimes there will be a bit of an abrupt transition between two versions of iOS that’ll cause headaches. Sometimes the compiler doesn’t tell you that what you wrote only works on the latest version of iOS. Suddenly your app crashes during the testing period and you’ve discovered that everything you wrote won’t work on older versions. I’ll use the example of a UIAlertController which was introduced in iOS 8. I wanted to support iOS 7 and 8 simultaneously and had to make a choice whether to use UIAlertView only (risking future deprecation) or use NSClassFromString(“UIAlertController”) to check for nil and write both. At some point I knew that UIAlertView would become fully deprecated and stop being supported, so in the interest of future proofing my app I chose to support both. This involved writing a small amount of code to support iOS 8 and a lot of code (including delegate methods) to support iOS 7. Now that iOS 9 is out and I’m supporting iOS 8 and 9 exclusively, I was able to go back into the project and remove all the offensive iOS 7 code, eliminating a bit of bloat in the process. Again, this doesn’t always happen, but the differences between the iOS 7 and 8 SDKs seemed much more stark than the comparatively easy transition between 8 and 9.

Selecting a Development Language

Another consideration is determining which language to develop with. Traditionally iOS apps were written in Objective-C, which is still widely used. As of iOS 7, many developers have chosen to adopt Swift, Apple's newer development language. While I personally enjoy Swift a good bit more than Objective-C and find myself writing much fewer (and cleaner) lines of code as a result, it is not without its growing pains and the changes between each language update can be quite drastic. This can lead to hours of code changes and problem solving just to get the project to compile. Additionally, when Swift was first introduced, if you chose to use it as your development language, you limited yourself to supporting only the latest version of iOS.

Potential pitfalls aside, I've chosen to use some of the latest technologies in my latest projects and have come to the conclusion, based on high iOS adoption rates, that I'd be reaching the vast majority of iOS users if I support the two most recent major operating system versions, iOS 8 and 9. From what I've seen, the prevailing wisdom seems to be that you really can't go wrong if you support iOS for the prior year. Apple's faithful supporters tend to be an eager bunch and will upgrade if they can, unless there's a really good reason not to, which rarely seems to be the case. You can be confident that you'll still be reaching the vast majority of your audience.

ABOUT THE AUTHORS

David Diehl *has been developing software professionally for over 15 years for a wide range of customers in different environments. He currently focuses on iOS development using native technologies and is passionate about creating elegant solutions for complex problems.*

Geoff Bender *began his programming career as a ColdFusion developer back in 2000 by writing and maintaining distance learning web applications for VCampus Corporation. He worked as a contractor to the General Services Administration (GSA) from 2001-2005 and redesigned several websites for the Chief Financial Officers Council, Chief Information Officers Council, and several other executive agencies for which he received recognition from the Executive Office of the President. From 2005-2007 he created energy analysis software for Pace Global Energy Services and Gazprom. Since 2007 he has worked as a senior ColdFusion developer for Segue Technologies on the Unites States Air Force's MPES project and has doubled as Segue's lead mobile developer for Apple's iOS platform.*

Mary Lotz *is Segue's Director of Engineering. She is a certified project manager (PMP) and scrum master (CSM), and has been directing application development teams and projects for a variety of customers and industries for over 15 years. A former application developer, Mary holds both B.S. and M.S. degrees in Applied Computer Science.*

Mark R. Andrachek, Jr. *has been a developer for over 20 years (5 years with Segue Technologies, Inc.) with experience building everything from small web pages to architecting large scale enterprise applications for customers in Education, Banking, Non-Profit, and Government markets, utilizing a broad range of platforms and technologies. His Java and Linux experience made Android just the right fit for his entry into mobile, where he's led the way for Segue since the creation of our first Android application in 2010. In addition to his mobile work, Mark has continued to develop and support Segue's efforts on the USAF MPES project. He lives in Crewe, Virginia with his wife, 3 children, three cats, and a dog.*

Matthew Gorbsky *has been designing commercial and enterprise solutions for web and mobile applications in the healthcare IT, financial services, Non-Profit, federal government, and commercial industries for over 10 years. He has also been working as a Certified Scrum Product Owner and Certified ScrumMaster to bring new products to market quickly.*

Ron Novak *leads the Commercial and Non-Profit customer verticals as a part of Segue's leadership team with the guiding principles of innovation, exceptional customer support, and customer value. His diverse experience in Information Technology has been formed through a consistent desire to understand the total picture with respect to information exchange, security, and the end-user experience. Ron is currently focused on leading development of Mobile Technologies to help businesses reach their customers in exciting new ways and supporting Non-Profits in better serving their members. Mobile application development is a natural extension of Segue's capabilities in Web development; Ron is applying this Segue experience to rapidly develop exceptional applications at low cost. Under his strategic direction, Segue's Mobile Application Development Team has grown from a small two-person operation to a cohesive development team that is releaing capability-specific mobile apps for Android and iOS platforms.*

This eBook contains content provided by Adam Zolyak, a former employee of Segue Technologies.

Thank you for reading our eBook. We hope you found it informative and interesting. If you are interested in working with Segue, please contact us so we can learn about your needs and plan a development path that works for you.

Segue Technologies provides:

Web
- Custom Application Development
- User Interface Design
- Content Management Systems

Data
- Business Intelligence
- Data Quality and Profiling
- Enterprise Data Management

Mobile
- iOS Development
- Android Development
- Mobile Web