Mobile App Planning Guide

by Optimus Information

If you're looking to develop a mobile app for your organization, this guide will walk you through what you need to know in order to make it successful while avoiding common pitalls.

Optimus Information is a mobile application and website development firm located in Vancouver, Canada. Visit our website at www.optimusinfo.com to get your mobile project started today!

eBook

optimus information

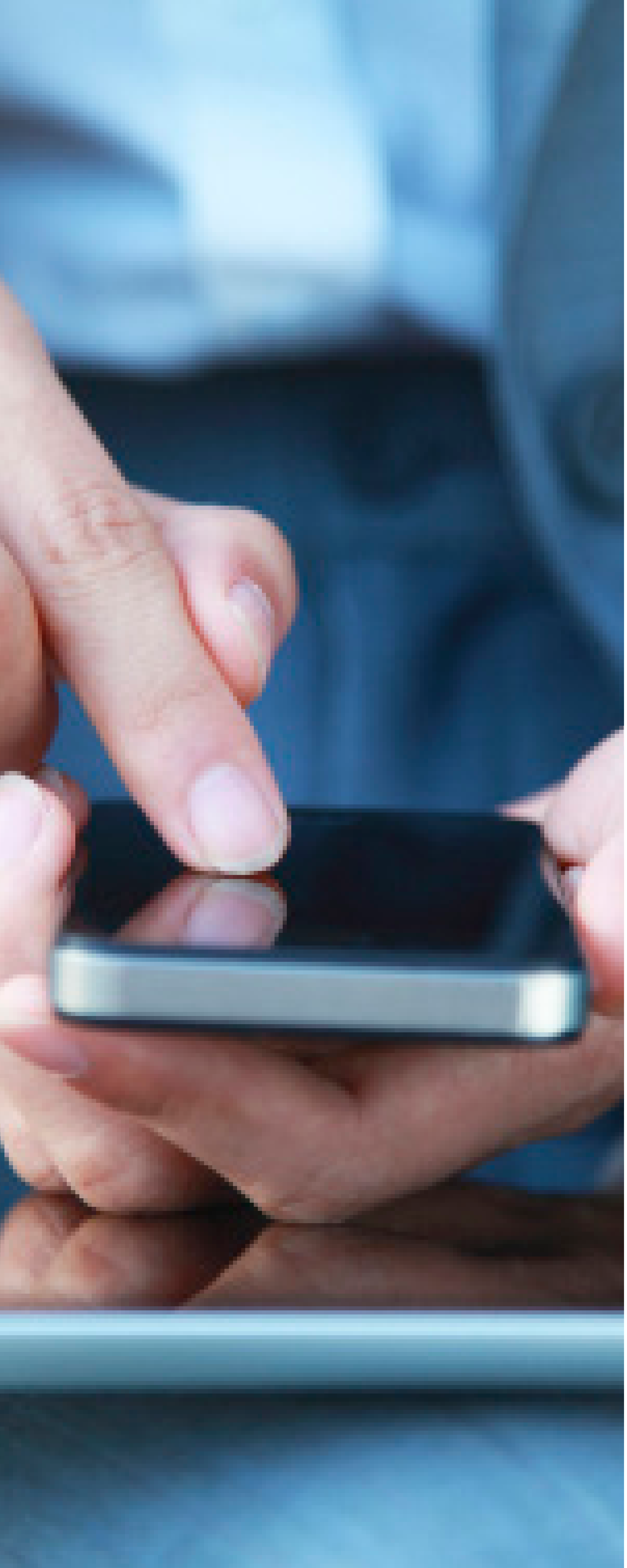


Table Of Content

Introduction

This guide will help you take your mobile app idea from concept to requirements that you can hand to a developer.

It will help you communicate with developers and ensure that developers aren't guessing at potentially important details.

It will help you reduce the time spent trying to figure out what are the obvious and non-obvious needs and communicate those needs to developers.

Who should read this guide?

This guide is primarily for product owners at independent software vendors tasked with creating large or complex mobile apps.

It should also provide value to anyone planning or considering a large or complex mobile app.

How To Use The Guide

As a product owner, you will need to be familiar with everything in the guide.

Each section has a deliverable and enough basic information to understand it at a high level and ensure that you can hand off the deliverable to the right person. Many of the sections have recommended resources that are there to help you better understand the deliverable should you desire.

Some of the sections won't apply to your app. Skip them.

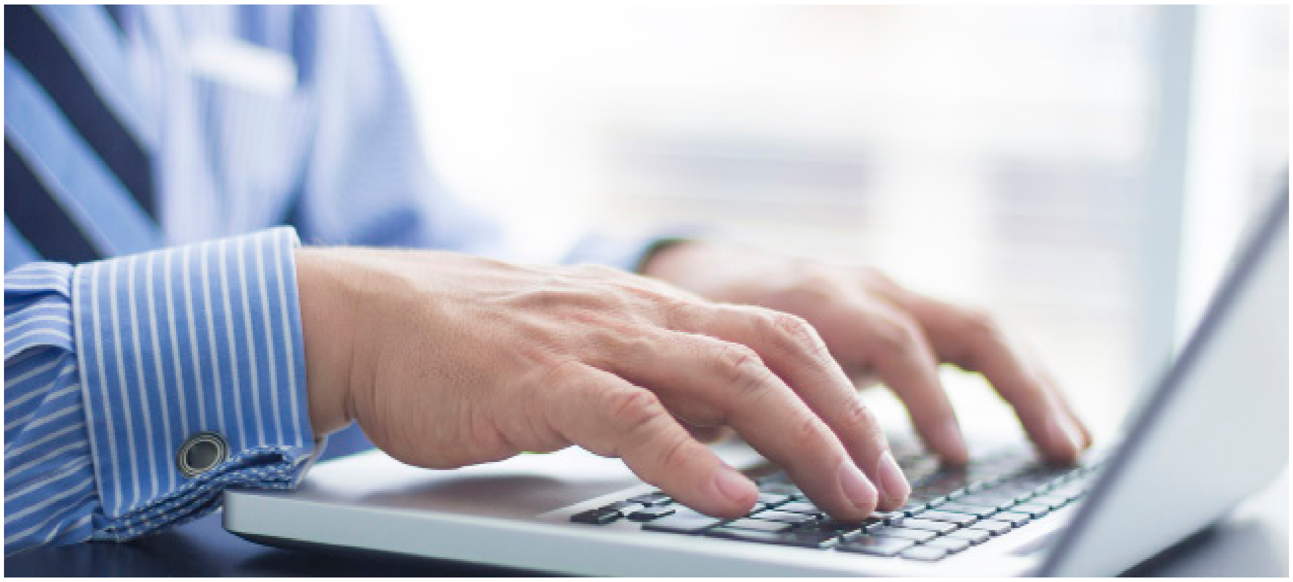
The deliverable samples are kept deliberately simple so that you can provide enough direction and oversight without a lot of specialist knowledge.

Some of the deliverables may require specialist knowledge that you will get from departments like marketing, development and legal.

If specialists on your technical team insist on adding a whole bunch of details that aren't set out in this guide, let them.

This guide is meant to help communicate with your development team. If you are struggling with any part of the guide, then communicate with your development team. You don't need to finish everything here to start a dialog.

Pre-Planning



You have decided to build an app. You have done some research and maybe even tried validating your concepts.

Now you need to take your idea from napkin requirements to something that developers can actually deliver on.

WHY ARE YOU BUILDING THE APP?

Apps can be made to generate money or awareness, to improve business processes or for a myriad of other reasons.

Make sure you are clear about the desired outcomes, so the development team is able to keep these outcomes in focus.

DELIVERABLE: 3-5 SMART GOALS

SMART stands for specific, measurable, attainable, relevant and time-bound.

Here are some examples.

1. At least 25 percent of users need to use the app an average of once a day.
2. At least six percent of users need to upgrade to the premium version within a month after installing.
3. The mobile app needs to generate enough ad revenue to cover 10 percent of development costs within three months of launch.
4. The number of house calls that our field workers can make in a day needs to increase by 10 percent one month after adopting the app.
5. The mobile app needs to increase the frequency of engagement with our web application by 20 percent within three months of rolling it out to customers.

WHO ARE YOU BUILDING THE APP FOR?

If there is no value in the app for the end-user, then there will be no value in it for you.

Make sure you know what problem you are solving for the end user.

Be prepared to answer detailed questions about the end users or, if possible, to give the development team direct access to them for requirements gathering.

DELIVERABLE: USER PERSONAS

User personas help designers and developers keep their focus on the end-user of your application.

Here are some examples:

KARL	SUE
Karl always rushes to get his work done and his attention to detail suffers. He tends to stick to technologies that he understands and resists new ones until he gets comfortable with them.	Sue is responsible for dispatching workers on-site and documenting all site calls. She also collects and files all paper-based work from the on-site workers and logs each call in a spreadsheet.

WHAT DEVICES ARE THEY USING?

Now that you know who your customers are, it is time to figure out what devices they are using.

And that means market research. But you don't need expensive focus groups when there are plenty of resources that can get you good information at a fraction of the price.

Try the following:

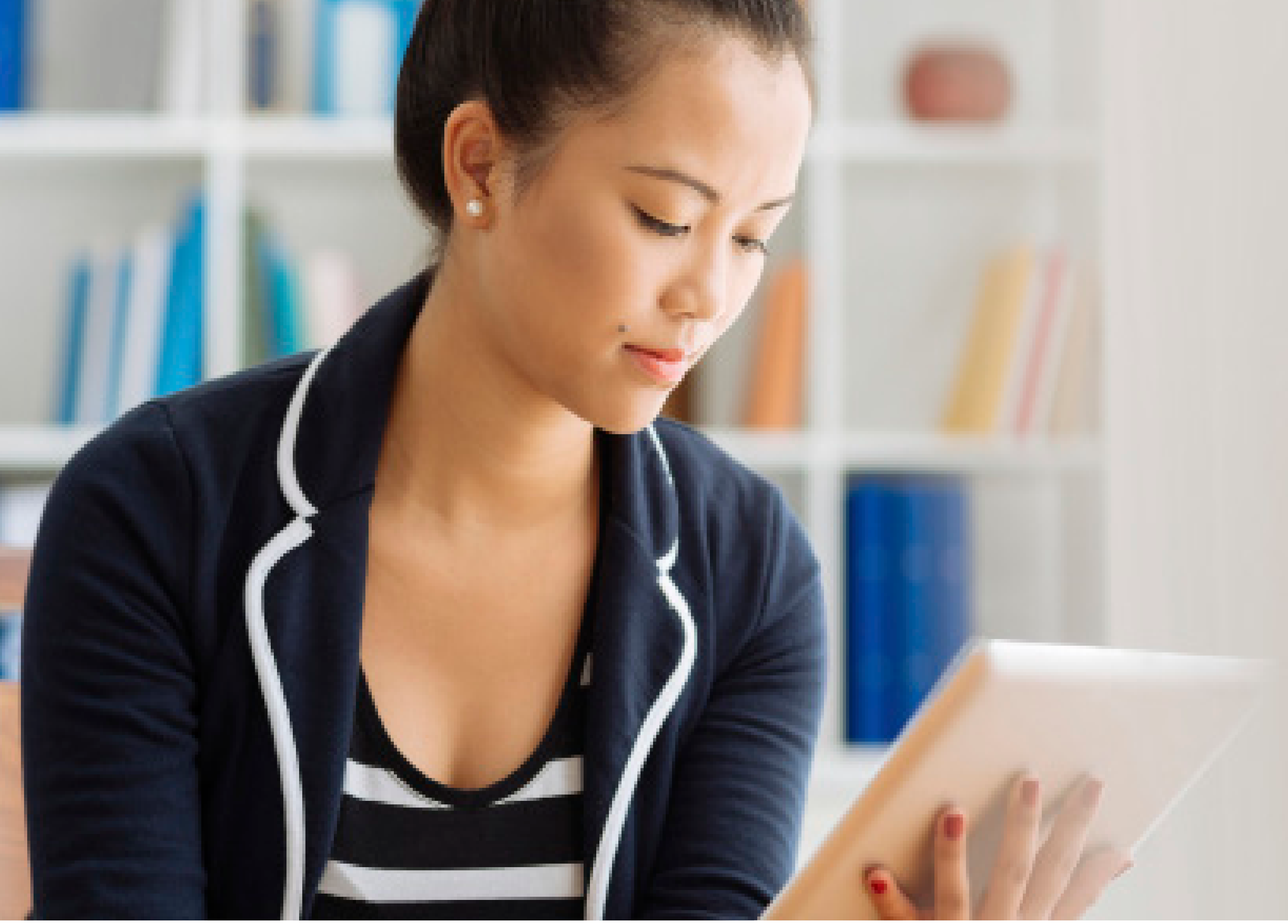
- Looking at your web analytics.
- Conducting online surveys.
- Asking customers directly.
- Checking free or paid statistics online.

DELIVERABLE: PLATFORM RANKING

Rank the platforms in order of importance including the OS version. The recommended resources below will help you choose OS version.

Here are some examples:

- The app will support iOS 4.0 and up first.
- The app will support Android 3.3 and up second.
- The app will Windows Phone 7 and up third.



WHAT DOES YOUR DREAM APP DO?

Don’t worry about expense just yet. Just start thinking about everything you want in an app and get others to do the same.

DELIVERABLE: PRODUCT BACKLOG

Start capturing your good ideas in a product backlog and then prioritize them. Encourage others in your organization to share their ideas and include the best input. You can keep adding ideas as you go, but it is important to start capturing them.

Your development team may already have a project management tool for capturing features in a product backlog. If that’s the case then use that.

Otherwise, a spreadsheet with minimally Title and Description and optionally Owner and Business Value will get you started.

WHAT IS YOUR BUDGET AND WHAT ARE YOUR ORGANIZATIONAL PRIORITIES?

With a limitless budget, you can get everything that you want. But in the real world, you will need to make compromises.

Develop a clear picture of what you want, then understand your organization priorities, so that you know what to sacrifice to get under budget or to fall within a timeline.

DELIVERABLE: SCOPE, BUDGET, TIMELINE RANKING

With the backlog filling up, you need to think about the relative importance of scope, budget and timeline and rank them so that you can decide what gets done and what can wait.

Also be clear on what elements are non-negotiable.

Here is an example:

Our timeline is strict and our budget is tight, so compromises will mostly need to be made in scope.

IN-HOUSE OR OUTSOURCE

Choosing whether development happens in-house or gets outsourced is important to the project's success.

In very broad strokes, the decision comes down to the following:

- If there is long-term work for the project team, then hiring staff to do the project in-house makes sense.
- If the project needs to be done as soon as possible or requires a large number of new developers, then outsourcing the project makes sense.

However, both in-house and outsourced projects have a number of advantages worth considering and disadvantages that can be mitigated.

What Does Your App Need To Do?

(Functional Requirements)

You are capturing ideas in your product backlog. Now it is time to turn those ideas into high-level requirements and then start drilling down into user stories.

HIGH-LEVEL REQUIREMENTS

What basic set of features is needed to make a useful app? What are the current architecture, skills and software licenses already in use at your company that drive architecture requirements?

These are your high-level requirements.

DELIVERABLE: HIGH-LEVEL REQUIREMENTS

High-level requirements shouldn't take more than a few sentences or bullet points.

Here are some examples:

- Sales staff need to be able to make updates while out of the office.
- Management need to be able to check sales performance from anywhere.
- HR needs expense reports.
- The backend should run on Azure and interface with our existing Azure systems.

USER STORIES

User stories facilitate communication between product owners and the development team. They capture the essence of your requirements in a way that lets developers choose the best approach for delivering them.

User stories are written in the following format:

- As a (user role), I want (function) so that (benefit).

DELIVERABLE: USER STORIES

Your product backlog is going to be the source of your user stories, but you only need to write user stories for features that you expect to implement in the short term.

This is how you should go about creating them:

1. Pick the biggest items in your backlog and write a user story for them. Sometimes you will see these larger user stories referred to as Epics.
2. Start breaking down any critical big user stories into smaller, more detailed user stories.

Write user stories for important smaller items in your backlog that you expect to implement.

The developers will be responsible for estimating user stories.

Once the developers have estimated the scope of each user story, you are then in a position to decide what gets done and what gets postponed.

Non-Functional Requirements

Mobile devices have a lot more constraints than web applications making the non-functional requirements for a mobile app even more important than for a web application.

In many cases, you have to make compromises as choosing to prioritize one non-functional requirement works to the detriment of another. It is worthwhile to rank your non-functional requirements in order of importance so that conflicting requirements are treated with the proper weight while setting out an acceptable service level range.

Non-functional requirements can be separated into the following categories:

- Availability and Robustness
- Instrumentation
- Performance
- Extensibility
- Scalability
- Security
- Privacy
- Data Integrity
- Serviceability
- Maintainability
- Migration
- Localization
- Data Validation
- Device Support
- Offline Support
- Analytics
- Testing

AVAILABILITY AND ROBUSTNESS

How reliably should users be able to access the system?

Availability is particularly important for apps with backend systems as it will affect the architecture and costs.

DELIVERABLE: SERVICE LEVEL AGREEMENT

If the app is running on your existing backend, you may be able to use your existing SLA with minimal modification.

Here are some examples:

- The system must be accessible 24/7 with 0.999 availability during that time.
- The system must be accessible during business hours from 7 AM PST to 9 PM PST with 0.99999 availability during that time.

INSTRUMENTATION

Do you need any code added to the system for the purposes of monitoring and measuring the product during program execution? Instrumentation requirements will be the responsibility of your CTO or whoever is responsible for the long-term smooth functioning of the app and its related systems.

Information from instrumentation may be output to a log or to a range of profiling tools.

DELIVERABLE: INSTRUMENTATION REQUIREMENTS

Specify what user stories need instrumentation and where to output this information.

Here are some examples:

- The Login, Add to Cart and Checkout user stories should be logged in Visual Studio.

PERFORMANCE

Minimally, you should specify what kind of initial screen load time and response time users should expect.

In most situations, initial load time should be under two seconds at the slowest.

Consumer-facing mobile apps should at least have the illusion of being instantly responsive.

If a the app involves interaction with backend systems, then specifying workloads is relatively easy to estimate without technical knowledge.

For more complex projects, your tech team may want to specify performance metrics for backend systems like throughput, latency and CPU benchmarks.

DELIVERABLE: PERFORMANCE REQUIREMENTS

Here are some examples:

- The app should load in under 1.2 seconds under normal circumstances and be instantly responsive.
- The backend must support 120,000 a day.
- These users will complete the Purchase workflow (Home -> Category -> Product Detail -> Login (60%) -> Add to Cart ->

EXTENSIBILITY

What sort of future growth or changes does the application need to accommodate?

DELIVERABLE: EXTENSIBILITY REQUIREMENTS

Here are some examples:

- The product classes will need to be extended to include product videos.
- Nutritional information is expected to be added to individual menu items.
- The application will allow new forms to be created using standard form elements.

SCALABILITY

How will the system scale as usage grows?

DELIVERABLE: SCALABILITY REQUIREMENTS

Here are some examples:

- The application will be hosted on AWS, so we'll just throw more EC2 instances at any problems.
- The application needs to support up to three additional servers in two locations to support anticipated future loads.
- The application is not expected to need scaling.

SECURITY

Are there any security requirements particular to your application?

Basic security best practices like not storing and sending unencrypted passwords over the web should be a given for any app. That is unfortunately not always the case in practice, but we will assume that you are working with good developers and you don't need to specify basic security requirements.

Security requirements may be driven by laws in financial, medical and other highly-regulated industries. Be sure to specify these where they apply.

DELIVERABLE: SECURITY REQUIREMENTS

Here are some examples:

- The application must be SOX-compliant.
- The application will validate forms against possible code injection.

PRIVACY

What end-user data will the application access? How will that data be handled?

DELIVERABLE: PRIVACY REQUIREMENTS

Here are some examples:

- The application will only ask for the minimum information required to run.
- The application will ask for name and contact information that will be saved to our CRM.
- The application will encrypt personal data for transmission over the web.

DATA INTEGRITY

Depending on whether your mobile app has a backend component or not, the following questions cover data integrity requirements for mobile apps.

Should the mobile app trust data from the backend?

What measures will be taken to ensure that data is accurate and reliable through transfer, storage and retrieval despite hardware failures, software bugs and human error?

For how long should stored data be retained?

Data integrity requirements can be driven by laws in 7nancial, medical and many other industries. Public companies also may have legal obligations that will drive data integrity requirements.

DELIVERABLE: DATA INTEGRITY REQUIREMENTS

Here are some examples:

- Financial data should be retained for seven years for SOX compliance.

SERVICE ABILITY

How will the application be distributed and updated?

For consumer-facing apps, this will typically be through app store default mechanisms. There are different platform-specific mechanisms in place for internal company apps as well as platform-agnostic mechanisms through mobile device management tools.

DELIVERABLE: SERVICEABILITY REQUIREMENTS

Here are some examples:

- New versions will be handled through the default mechanisms in the Play Store for Android and App Store for iOS.
- New versions will be handled through the iOS Developer Enterprise Program in-house distribution mechanisms.
- New versions will be distributed through the Windows Phone Company Hub app.
- New versions will be distributed through AirWatch.

MAINTAINABILITY

What coding and development standards will be required to ensure long term maintainability of the software.

Coding standards can include the following:

- Architecture
- Comments
- Code review

Development standards can include the following:

- Source control system
- Issue tracker
- Documentation

The issue tracker and source control system will likely be whatever your technology department already use.

DELIVERABLE: MAINTAINABILITY REQUIREMENTS

Here are some examples:

- Code will be commented as per industry standards.
- All methods will have headers.
- Code will follow standards set out by Android, Apple and Windows Phone.
- All code will be checked into SVN.
- All tickets will be tagged to Redmine issues for traceability.
- The design and code will be continuously monitored for modular and maintainable implementation.

MIGRATION

Does the application require data to be migrated to allow for mobile access? If so, what needs to be migrated?

And do you have a preference to where the data gets migrated?

DELIVERABLE: MIGRATION REQUIREMENTS

- CSV data needs to be transferred to a yet to be decided database.
- Legacy data stored in Oracle needs to be transferred to our MSSQL production environment.

LOCALIZATION

Does the application need to be structured to support multiple languages?

If so, which ones? Pay particular attention to non-Latin alphabets.

DELIVERABLE: LOCALIZATION REQUIREMENTS

Here are some examples:

- The application will be structured to support multiple Latin languages.
- The application will be structured to support Latin languages, Chinese, Japanese and Korean.

DATA VALIDATION

What measures will be taken to ensure data entered in to the application is valid and correct?

Where will that validation occur? The application? The server? Or both?

How will invalid data be handled?

DELIVERABLE: VALIDATION REQUIREMENTS

Here are some examples:

- The application and server will check user input for validity.
- Input that is not valid will return a descriptive error to the user.

DEVICE SUPPORT

What devices will you support?

How should device compliance be determined? manually on emulators? manually on one physical device? automatically on many devices through a device cloud?

The Recommended Resources section includes good sources for this information. Your web analytics are also a good resource.

DELIVERABLE: DEVICE SUPPORT REQUIREMENTS

- Android will support version 2.3.3 and up.
- Compliance will be checked on manually on the devices in the device library.

OFFLINE SUPPORT

How should the application behave when it loses connectivity?

Should it return an error? Or save the data and sync it later?

DELIVERABLE: OFFLINE SUPPORT REQUIREMENTS

Here are some examples:

- Users will be offline for long periods of time. Workers should be able to add new pictures, video and text for four hours before having to sync systems.
- Progress should be saved locally when connectivity is lost. An error message should alert the user that the connection has been lost, but their progress has been saved.

ANALYTICS

What mobile analytics options should be installed?

Google Analytics and Flurry analytics are two popular choices and a good place to start if you don't already have a choice.

DELIVERABLE: MOBILE ANALYTICS REQUIREMENTS

Here are some examples:

- The application will use Flurry analytics.

TESTING

There are a number of unique challenges when testing mobile apps. The large number of devices with a wide array of screen sizes makes it a challenge to adequately test all of supported devices and screen sizes.

At the least expensive and robust end, testing can be done on emulators and whatever devices the testing team happen to own. The most robust, and expensive, option is to use device clouds that let you automate your testing on any devices that they have.

The option you choose will be a negotiation between your desire to release a robust app that delights end users and your budget.

DELIVERABLE: TESTING REQUIREMENTS

Here are some examples:

- The application will be tested on emulators and any conveniently available physical devices.
- The application will be tested on all supported devices on Perfecto Mobile.

User Interface

Unless you have your own user interface specialist, you won't be expected to come up with a complete user interface. But in order to maintain brand consistency, be prepared to give the following information.

FONT

- **Font stack:** ie. Helvetica Neue, Helvetica, Arial, sans-serif.
- **Heading format:** color, font stack and size of headings.
- **Link format:** color font and size of links.

COLORS

The exact template for colors that you use will depend on your branding. The following will cover most color requirements.

- **Primary:** primary branding color.
- **Secondary:** secondary branding color.
- **Background:** background color or image outside the main content area.
- **Main content background:** background color used behind text in main content area.
- **Alert:** color of any alert notifications.
- **Success notifications:** color of any success notifications.
- **Navigation:** color of navigation bar and fonts within the main navigation.

LAYOUT

Typically specifying layouts isn't necessary particularly for native apps which will use native interfaces that cover the many of the common layout elements. However, you may want to specify the following.

- **Smartphones and Tablets:** do you want a separate layouts separately optimized for smartphones and tablets.
- **Branding:** location and format of brand elements.
- **Navigation:** do you want persistent navigation.
- **Paging:** do you want infinite scrolling of lists or do you want to separate them into pages.

Type of App

There are three ways of building a mobile app:

- **Native:** applications built for the specific operating system they are operating in.
- **Web:** applications built in standard web languages that work in all standards-compliant mobile web browsers.
- **Hybrid:** applications built primarily using web technologies but with native elements.

Each approach has different advantages and disadvantages and the best approach depends on your goals, your end users and the requirements.

NATIVE

Native applications are built for a specific operating system in that system's native language.

- **iOS:** Objective C
- **Android:** Java
- **Windows Phone:** C#/.Net

Advantages	Disadvantages
• Full access to device features like camera, GPS and storage. • Support native compound touch gestures like double taps and pinch. • Best performance. • Native user interface. • Easier discovery thanks to app store distribution.	• Requires separate programming language for each operating system. • More expensive. • Relies on high-demand programming skills. • Maintaining and updating the app is also more expensive.

WEB

Web-based applications are built using standard languages like HTML5 and CSS 3 that are compatible with all modern smartphones.

Advantages	Disadvantages
• Less expensive to build and maintain. • Relies on high-supply programming skills.	• Minimal access to device features. • Limited offline storage. • Less secure. • Can't match native performance. • Non-native user interface. • No app store discovery.

HYBRID

Hybrid applications use the standard languages like HTML5 and CSS 3 to maximize reusability across all platforms with native code to access device-specific features that aren't available to purely web-based solutions.

Advantages	Disadvantages
• Nearly full access to device features. • Relatively easy to maintain and update. • App store discovery.	• Less expensive than multi-platform native development, but savings aren't nearly what you would expect. • Can't match native performance. • Non-native user interface.

Conclusion

The time and thought required to produce all of these deliverables is an investment in the future success of the app. It will help development go more smoothly, saving time and money, while ensuring that the app delivers the expected value.

The information you have collected will help drive the conversation with developers. If you are uncertain or struggling with certain details, then contact Optimus Information for a free one-hour consultation.