# **The essential Google Play Store app pre-release checklist**

**1. Put your app on a diet**

No one likes to download a large app, especially when they are on a slow network like 2G or Edge (Yes, it still exists.) It is highly likely that a user lands on your app’s page on Play Store, decides to install it, but then cancels it later just because it took ages to download.

An APK (Android Application Package) is simply an archive containing your app’s classes, resources, assets and several other files. You would want to keep it as small and tight as possible. There are several ways you can optimize your app size:

* Always use available tools (like ProGuard or DexGuard) to shrink the size of your app by removing a plethora of unused methods. This step itself can show you a significant reduction in app size. Also compressing the resources (like images, audio files, etc) included in the APK is a no-brainer.
* If your app needs to store several assets (like images, video clips, audio clips, etc.), then it is always better to off-load them to your server and load them on-demand.
* Before releasing your app, spend time analyzing what’s actually going on inside your APK. Android Studio 2.2 has an [APK Analyzer](http://android-developers.blogspot.in/2016/05/android-studio-22-preview-new-ui.html) built-in just for this purpose.

***Bonus****: Use tools like [TinyPNG](https://tinypng.com/) or [ImageOptim](https://imageoptim.com/) to significantly reduce image size, losslessly. You are welcome.*

**2. Make your app speak several languages**

This is probably the best time to prepare your app for localization. Your app will run on a variety of Android devices in different regions, and you absolutely want to make your app usable to everyone.

Suppose you have made a simple note-taking Android app supporting only your native language, English. But it doesn’t mean that a French or a Spanish user wouldn’t want to use your app or buy its premium version maybe. You can massively increase the target audience of your app by supporting multiple languages.

Before releasing your app, make sure that all the strings are properly placed in the *strings.xml*file and nothing is hard-coded anywhere in your code. Check [here](https://developer.android.com/guide/topics/resources/localization.html) for detailed guidelines on this.

You can either pay for professional translation services or find a bunch of interested users who would like to translate your app for free. You can gift them the premium version of your app as a token of appreciation.

***Pro tip****: Localization is not just translating. It’s much more than that.  Use tools like [Crowdin](https://crowdin.com/) or [Weblate](https://weblate.org/en/) to professionally manage your translators.*

*[ Get Report:*[*The Top 20 Continuous Application Performance Management Companies*](https://www.microfocus.com/en-us/assets/it-operations-management/research-in-action-continuous-application-performance-management-saas-and-software?utm_source=techbeacon&utm_medium=techbeacon&utm_campaign=00134846)*]*

**3. Time to perform an in-depth performance audit**

You are almost ready to release your app, but are you sure your app is performing at its best potential? You need to ensure that your app is fast, responsive and works flawlessly. The first impression is the last impression.

After months of hard-work developing your app, you need to put this extra effort and ensure that your app “makes the cut.”

It's the right time to start polishing your app and make it shine. Use [LeakCanary](https://github.com/square/leakcanary) to detect memory leaks quickly and easily. It is very likely that you have introduced some leaks in your app (even the Android Framework have a few) that need to be fixed. Always remember, even a tiny leak can sink a large ship.

[Here](http://blog.nimbledroid.com/2016/05/23/memory-leaks.html) is an excellent article explaining how you might have introduced memory leaks in your app and how they can be fixed. Ensure every heavy operation (file, database, network, etc) is off the main thread. You want your app to be as responsive and as fluid as possible, right? You just have 16ms to render a frame, so be quick.

A thorough audit can reveal lots of performance issues in your app before releasing. Fix them and always ensure that you are delivering a top-notch quality app.

***Bonus****: Want to deep dive and learn concurrency in Android? A good start can be found*[*here*](https://www.packtpub.com/application-development/asynchronous-android)*.*

**4. Don't miss your security audit**

Performing a complete and in-depth security audit of your app is something you cannot afford to miss. And this is something better done before release, as there will always be a few things you cannot fix once your app is out in the wild.

If your app doesn’t deal with any sensitive user data, then you can probably skip this, but if it does then give me your undivided attention.

* Ensure that all the sensitive data is encrypted before storing them locally in the database, shared preferences or a file. All network transfers should be SSL encrypted. Never ever make the mistake of storing sensitive data in plain text locally**,** as it can easily get compromised if a hacker gets hold of the device.
* Also, ensure that passwords or pins are not encrypted, but hashed. There is a difference between hashing and encryption, and you need to learn their proper scope.
* Do not forget to obfuscate your code using a tool like ProGuard. You probably won’t want to leave your code wide open for anyone to see and analyze (unless you plan to make it open source)

***Pro tip****: ProGuard can protect your app, but its sibling [DexGuard](https://www.guardsquare.com/dexguard) does a far better job of it. It can not only obfuscate your code but even encrypt your classes and strings. DexGuard can also use reflection for making method calls thereby making it immensely difficult for an hacker to understand your app’s flow.*

**5. Protect your app from tampering**

Hackers can always try to tamper an app by reverse-engineering it in every way possible. Theoretically, it is impossible to hack-proof an app, but you can do your part and make the hacker’s job far more difficult.

The best time to take steps is just before releasing your app. If you do these later in a subsequent update, hackers can still get hold of a previous APK of your app and exploit loopholes from the older versions.

There are several things you can do to prevent tampering (to some extent), like preventing your production app to run on an emulator. It is almost impossible that a real user will use your app on an emulator.

You can make your app silently fail if it is debuggable (release APK and debuggable, sounds fishy?) or if it’s certificate doesn’t match with your signing certificate. The term “silently fail” is important here. Don’t let the hacker know that you are trying to prevent a tampering attempt.

Put highly sensitive business logic of your app in [Android NDK](https://developer.android.com/ndk/index.html), making it significantly difficult for hackers to understand and crack it. I am not claiming that NDK code can’t be cracked, but its far more difficult than plain old Java code.

***Pro tip****: Check out*[*this*](https://www.airpair.com/android/posts/adding-tampering-detection-to-your-android-app)*detailed guide on tamper detection. And this library, which can make your life a bit easier.*

**6. Get your crash reporting tool ready**

If you haven’t integrated a crash reporting tool in your app before releasing it yet, do it now. The last thing you want is to release your app into the wild and not know anything about what’s happening with it. If your app crashes, you as the developer should have every possible information about it so that you can take the necessary steps to fix it immediately.

I recommend [CrashAnalytics](http://try.crashlytics.com/). I have been using it for more than a year now and am completely satisfied with it. I have used [Firebase Crash Reporting](https://firebase.google.com/docs/crash/) tool as well, but won’t recommend it right now as it is in beta and there are some serious issues with it.

Also remember to log the caught exceptions explicitly. A caught exception is still an exception and you need to know everywhere it's happening in your app.

There are some crashes or exceptions which only occur on a particular device or a particular Android version and you need to specifically deal with them.

A good crash reporting tool is like a good friend who will constantly help you and guide you to get better with time.

**7. Test your app on different devices**

You probably have been using one or two devices to develop your app and have thoroughly tested on them. You have squashed all bugs and crashes and is ready to release, but wait, it’s just the tip of the iceberg.

You need to test your app thoroughly on various Android platforms and also on various device models from at least some popular manufacturers like *Samsung, LG, HTC, Motorola*who hold a considerable amount of market share in Android.

There are several minor to major API changes in every version of Android which needs to be dealt specifically. If you are doing heavy file-handling, then you must test properly on various platform versions (remember [what](https://plus.google.com/+TodLiebeck/posts/gjnmuaDM8sn) Google did with KitKat) and also on devices from various manufacturers.

Almost all OEMs customize their ROMs in an attempt to create an unique experience for their users which makes it even more important to test your app on them before making a final releasing.

Buying a lot of devices for an individual developer can be a real pain. I wish I had an arsenal of devices to test on, but Santa keeps on ignoring my letters.

***Bonus****: Testing on emulators like [Genymotion](https://www.genymotion.com/) can help you cover the basic tests to some extent, but there is nothing that can replace a real device.*

**8. Plug in your favorite analytics tool**

With your app released in the market, you would want to know how your users are using the app, which features of your app are the most used and which are the ones completely ignored (and how you can improve them).

You need to know which screens users are spending most of their time on and where they are absolutely not. Armed with these valuable pieces of information, you can decide where to spend your time, effort and money on. You can effectively decide the future game plan of your app.

There are several analytics tools out there, both paid and free. But the free ones are more than enough for small, mid-scale and even some large-scale apps.

***Pro tip:****Try*[*Firebase Analytics*](https://firebase.google.com/docs/analytics/)*. It’s completely free with unlimited event tracking, really cool isn’t it? I have also used and have been a big fan of the real-time analytics offered by*[*Answers*](https://answers.io/)*from Fabric.*

**9. Make a proper release plan**

You have worked so hard on your app for months and you don’t want to screw your app release out of hurry, right?

Don’t just go ahead and put your app on the Play Store immediately.Spend some time in making a proper release plan. First, make a small Alpha release of your app to some of your offline friends.

You can directly observe how these users are using your app, where are they getting stuck, what initial impressions are they having. Make a proper mix of both techie and non-techie users to have much better results.

After a few small alpha releases and initial improvements, you can go ahead with a beta release. This can and should be big.

Try out the [open beta release system](https://support.google.com/googleplay/android-developer/answer/3131213?hl=en) recently started by Google Play. With a beta release, you will have a much bigger and better insight of the state of your product.

Various users from various countries having various devices will be using your app and this is *exactly* what you want. Get ready to witness a lot of bugs and crashes. But in the end, you will get some invaluable feedbacks which will help you to significantly improve your app.

Now your app is ready for a final release. You can be quite assured that no major issue can come up now. You can even refine the experience of your final release, by opting for a staged rolloutwhere you push the final version of your app incrementally to small portions of your users.

***Pro tip****: Try the*[*beta*](http://try.crashlytics.com/beta/)*by CrashAnalytics to seamlessly manage and distribute your beta app.*

**10. Decided on your app’s minimum SDK**

It’s the right time to finalize the minimum SDK you are going to support. It basically depends on the features your app wants to support and the user base it wants to target. There should be a proper balance between these two.

If your app is using several non-backportable Material Design APIs, then you have to keep your minimum SDK to 21 (Lollipop). You wouldn’t probably want this as only 40% of the current user base is using Lollipop and above. You don’t want to miss out on such a huge volume of potential users, right?

Spend time on backporting features if you haven’t done it yet. If you decide to support API level 19 (Kitkat) and above, **y**our potential user base jumps significantly from 40% to around 75%.

But my personal advice would be to support API level 15 and above (if possible) which lets you tap into above 97% of the current Android user base.

**11. Make way for easy user feedback**

Once your app is released and your users start using it, they will face some issues or have some suggestions and would like to get in touch with you directly.  It is very important to implement a proper feedback mechanism in your app.

Try to place the feedback action somewhere in the Navigation Drawer or in the Toolbar, so that it is easily accessible to your users.

Also consider adding some FAQs in the feedback section which can solve some of the common queries almost immediately. This will not only save your users some valuable time, but will also save you from attending lesser support mails.

You should also give access to a public issue tracker (in [Github](https://github.com/google/google-java-format/issues)maybe) so that your users can list down the bugs that they discover and it is also visible to all other users of your app. It is very important to be transparent to your users.

**Bonus**: If you want an enterprise level feedback solution and can afford it, try [UserVoice](https://www.uservoice.com/) or [ZenDesk](https://www.zendesk.com/).

**12. Start investing time on App Store Optimization**

Haven’t heard of the term App Store Optimization (ASO) yet? It’s the right time to start learning about it. ASP is all about how you can optimize your app and your app’s listing on Play Store to attract organic downloads from the store.

Start preparing some clean and beautiful promotional graphics for your Play Store page. It is even better if you can record a small video showing off some of the coolest features of the app.

I have seen many developers making the mistake of writing their Play Store page description just before releasing their app. Don’t do it. Spend a considerable amount of time and effort researching how you can write an optimized description which can drive organic downloads. But don’t overdo it, or make it look spammy.

Spend time researching on the keywords you are going to target and how you are going to incorporate them in your page’s title and description. Thoroughly research who your competitors are and what they are doing to improve their rankings?

Rome was not built in a day. You cannot get your app optimized and ranked well within days. It takes time and money too, but once you get it right, you are on the path to reach your next few million users**.**

***Bonus****: Use services like [SensorTower](https://sensortower.com/) or [AppAnnie](https://www.appannie.com/) to get a complete insight into your competition and how you can improve your Play Store ranking.*

**13. Decide on a proper release date**

This is important. Plan a proper release date so that you or your team is available for the next few daysafter the app is released.

I had a bitter experience of making a big mistake when I had released a major bug fix update on a Friday and went on to a vacation for the weekends. When I came back on Monday, I was greeted with lots of 1 star ratings and emails regarding a crash that all users of a particular Android version were seeing. The issue was minor, but the impact was major. And it did hurt a lot.

This is something that you should religiously follow not only for your first release, but for every major update of your app. Always be prepared for contingency.

**14. Give it that finishing touch**

If you are taking care of all the points mentioned above, you are mostly good to go. But still, there are some minor things that you may be interested implementing to give your app the last finishing touch.

Before making the final release of your app, ensure all the to-dos have been addressed. We developers have a habit of adding to-dos whenever there is something we need to look into later, or whenever we get a bit lazy.

It’s also the right time to find a way to monetize your app. If you are good at something never do it for free. Either add in-app purchases or incorporate some ads (native ads, interstitial ads, etc) to start making money out of your product right away.

It is also a very good practice to add a changelog so that your users know exactly what have been fixed or improved from the previous version. It is also a good practice to credit and add the license information of all the open-source libraries you are using.