App Launch Checklist

~~Update Librari~~

~~Documen~~

~~Tests~~

~~Revie~~

BUGFENDER

Mobile App Launch Checklist

We created this resource to solve one basic problem: Once you go through all the work of building an app, how can you be sure to nail the landing?

In the rush to get an app out, many developers can forget to tag releases or lose depository information that simply needed to be earmarked for a future update. There are some final security measures that need to be taken to protect your app. Whether it's due to excitement or fatigue, it happens. Following a protocol now will save you a lot of time later on down the road.

On the user side, the last thing you want is a simple but unresolved bug to be the first thing to greet new app users. User support needs to be available along the way in multiple forms so that you don't lose users minutes after they download the app.

Fortunately, as a team of experienced mobile app developers, we have created a tried and true checklist for releasing an app, which we now release to you. If a brain surgeon has to follow a protocol, a software engineer may consider looking for a roadmap for how to launch an app to avoid unnecessary problems later on.

Don't let your mobile app launch be an afterthought.

A checklist keeps the future user in mind. Make sure the stars align so that when a customer meets your app for the first time, it's love at first sight, not delete at first download.

For any release, ensure you have the necessary tools in place to provide support for your app. We suggest the following steps to delight your users and make your life easier:

- ☐ Install Intercom to provide chat with customer support
- ☐ Add a logging tool to your app:
    - ☐ Bugfender to view your application's logs remotely and capture crashes
    - ☐ CocoaLumberjack (iOS)/Timber (Android) to capture application logs
    - ☐ Crashlytics if you only require crash collection
- ☐ Check that all issues/tickets/tasks corresponding to the release are marked as completed or have been rescheduled to another version.
- ☐ Check that all dependency libraries are up to date:
    - ☐ If there are minor version updates available, update them.
    - ☐ If there are major version updates available, create a new task to ensure they're updated in the the next release of your app.
- ☐ Compile your app's code and check there are no warnings.
- ☐ Run a static code analysis and linter, and fix any issues detected.
- ☐ Run all tests (unit, integration, system, etc) and ensure they all pass.
- ☐ Test your application:
    - ☐ On the real devices that most commonly use your app. These may not be the latest devices, but you should ensure your users are able to use your app.
    - ☐ The latest versions of iOS and Android using emulators if real devices aren't available.
- ☐ If your application requires a database which needs to be updated with the new release of your app:
    - ☐ Delete the new test version of the application from the mobile device.
    - ☐ Install an older version of the application and enter some test data.

    - ☐ Upgrade the installation of the app to the latest version and check the data migration worked properly.
- ☐ Tag a version of the code repository with a version number.
- ☐ For iOS apps, retain a copy of the generated dSYM file.
    - ☐ For security purposes, make sure debug symbols are not included in the binary (there's a build setting for it). If this precaution isn't taken, it could be used to reverse engineer or attack the app.
- ☐ If you use Proguard/Dexguard on Android, keep a copy of the mapping file (mapping.txt).
- ☐ Notify the development team about the new release in order to:
    - ☐ Test the deploy and make sure no-one is having issues.
    - ☐ Make sure everyone is updating their local working repositories to the newer version.
- ☐ Distribute a beta first, followed by production. It's also worth considering a staged rollout.
- ☐ Documentation:
    - ☐ Publish an internal document containing a list of tasks completed and included changes from the previous version.
    - ☐ Ensure your application's documentation, help pages, and manuals are up-to-date.
    - ☐ Create a detailed changelog for the public.
- ☐ Update the app's description and screenshots on the App Store/Google Play store.
    - ☐ Tip: Fastlane is a good tool to automate taking screenshots and sending beta and production builds to the app stores.
- ☐ Create a production build and upload it to the app stores.
    - ☐ We recommend you keep a copy of the built file as well (ipa or apk). It will help your team to easily test the application at any version.

BUGFENDER

bugfender.com