Preparing a Flutter App for Release

Introduction:

In today's world launching a new app requires everything in place like proper labelling, passing security checks etc. Similarly, preparing your flutter app for release involves updating versions, configuring permissions, thorough testing and more.

Steps to setup an App for Release:

1- **Versioning:**

It is essential for updating and managing app versions.
In Flutter you can set the version number is pubspec.yaml file

```
Version number: pubspec.yaml file
Version format: major.minor.patch+build
```

Example

```
Current version: 1.0.0+1
Next version: 1.0.1+2
Update: 1.1.0+2
```

2- **Updating Icons and Splash Screens:**

Packages help customize app icons and splash screens.
flutter_laucher_icons package allows to set custom app icon while Flutter_native_splash is used to create splash_screen.

```
Custom app icon: flutter_launcher_icons
Splash actions:
Flutter_native_splash
```

These Visuals help in branding and User Experience.
To set up the app icon add the following to your pubspec.yaml file.

```
dev_dependencies:
 flutter_launhcer_icons: ^0.14.2
flutter_icons:
 android:true
 ios:true
 image_path: “assets/icons/icon.png”
```

Run the command using following

```
Flutter pub run flutter_launcher_icons:main
For splash screen , use flutter_native_splash package:
dev_dependencies:
 flutter_native_splash: ^2.4.4

flutter_native_splash:
  color: “#42a5f5”
  image: assets/splash.png
Run:
Flutter pub run flutter_native_splash:create
```

3- Configuring Permissions:

App permissions help in transparency and user security
For Android : AndroidManifest.xml
Example

```
<user-permission android:name=”android.permisson.INTERNET”/>
```

For iOS: Info.plist
Example

```
<key>NSCameraUsageDescription</key>
<string>We need access to your camera to take pictures.</string>
```

4- Signing the App:

Publishing apps on Google Play Store and Apple Store needs Sign-in
Sign-in authenticates the app, ensuring a reliable source.

For Android:

```
Generate Keystore file
Configure in build.gradle file
```

For iOS:

```
Manage through Xcode
```

5- Obfuscation:

Obfuscation helps in making code harder to read and reverse-engineer.
It helps to :

- Publish apps
- Deny unauthorized access

Flutter provides built-in functionality for obfuscation.

For Android

```
Use command: flutter build apk –realease –obfuscate –
split-debug-info=/<director>
```

For iOS:

Set-up some flags in Xcode

6- Testing and debugging:

Testing and debugging helps app run smoothly without crashes.

```
Commands Used:
  • Flutter test: Runs unit tests
  • Firebase test lab: Integration testing
  • CI/CD pipeline: incorporate testing
```

7- Building the release Version:

Release build helps meet the performance and standards of platform targeted.

Build for Android:

```
flutter build apk -release
```

Build for iOS:

```
flutter build appbundle -release
```

8- Submitting to App Stores:

Continuous Deployment (CD) automates the deployment of your app whenever the changes are made.
By integrating CD in your workflow, You Can:

- Automate tasks such as building, testing and deploying tasks using CI/CD tools such as GitHub actions, Bitrise and CircleCI.

For Android:

Use Google Play console to upload your APK or AAB,

Provide Meta Data and Submit for review.

For iOS:

Use Apple Store connect to upload your build , provide meta data and submit for review.