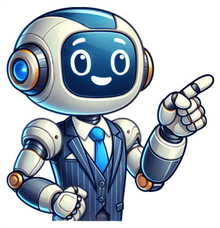
I'm not a robot



Circle to Search now recognizes contact details and links on your screen, and prompts you to act on them in the moment. Simply tap the chip to start a call, draft an email, or open a link without needing to copy and paste information into a new app. Action chips appear when a single phone number, email address, or URL is detected. Available on select devices. Internet connection required. Works on compatible apps and surfaces. Results may vary depending on visual matches. As per android official documentation: An adaptive icon can display differently depending on individual device capabilities and user theming. Adaptive icons are primarily used by the launcher on the home screen, but they can also be used in shortcuts, the Settings app, sharing dialogs, and the overview screen. In this short article, you can see a way to create and use adaptive icons using png or webp images, this also can be done using svg files, remember that for svg files you have to use a dedicated tool for designing the images. Finding the images You can use images from the internet for making the launcher icon for your android apps, in most cases, you can grab one from a external site, but, when using the file you must credit to its author. The images we'll be using are found in the Flaticon website, i like swimming, so the icons choosen. Flaticon is the largest free database of editable icons with over 7 million resources available. Themed icons With Android 13 (API level 33) or above, users can theme their adaptive icons, the feature can be enabled via device system settings. For the exercise, i selected the icons provided for the normal launcher icon and the themed icon. The icons used has to have equal size (512px in this case), both the launcher and the themed/monochrome icon has to be with transparent background. Add adaptive icons in android studio In this section, we go to android studio, we can choose to create a new project or use an existing one, just keep in mind that if the following actions are applied, it is recommended to keep in mind that when creating the adaptive icons they replace the existing icons (it is assumed there is version control applied in these cases). Once in the IDE, check the res folder, right-click on it and select the option New > Image asset. Once the image asset configuration dialog appears, we can select the image for the icon in the foreground layer tab, for this part we select the monochrome icon that we chose earlier, also, we set the scaling for the icon to be trimmed and to have a zoom of 60%. Later, in the background layer tab, we select the source asset type to color, and set the color using the hex color #FFFBFE, additional settings remain as the default in the dialog. After configured the icon, we do click Next, for reviewing the icon files to be generated. In this screen, we can see that the images are stored in the mipmap folders for each available app density, also, the images generated using the filenames ic_launcher.xml, ic_launcher_round.webp and ic_launcher_foreground.webp. At this point, we are going to use the ic_launcher_foreground.webp files, because those icons represent the monochrome foreground layer for the themed icon. The following step is to rename the ic_launcher_foreground.webp files to ic_launcher_monochrome.webp. for this, we can use the menu Refactor > Rename (Mayus + F6) option. After renamimg, the files are shown like this: Now, we repeat the steps for adding a launcher icon, this time, adding the normal icon we choosed. As result, we have both the monochrome and the colored foreground layer for the adaptive icon. Now, check the results for the adaptive icon display Now that we have ready the launcher icon using the normal colored image we used, this time we check the launcher.xml files, which contains the setting for the foreground layer and the background layer for the adaptive icon. In the exercise, we made both the not-round and the round launcher icon based on the configuration we used before for creating the launcher icons, so, we check both xml files: {root project folder}/app/src/main/res/mipmap-anydpi-v26 res/mipmap-anydpi-v26/ ic_launcher.xml ic_launcher_round.xml Both files have the following xml content for the adaptive icon configuration, in which is used the element to define the foreground, background, and monochromatic layer drawables for the icons: With that, the themed adaptive icon is configured. Now, let's check the result visually, for that, open the ic_launcher.xml file and ensure to have clicked the split option for viewing both the code and the design result. In the upper right side of the split view, we can see the icon with a representation for dark/night mode, when we click that option, we can see the options for selecting the theme mode (night/not-night) and the dynamic color theme. At this point, we can change the mode and the dynamic color for display the themed icon variant. Preview in devices After creating the launcher icon and the themed / monochrome variant, we can test the icon displayed in the device or the emulator, for that, just build and test the android app, for the device, try to use a device/emulator using android version 13 or above. From left to right: Displayed normal launcher icon in the home screen Wallpaper and style system configuration, here, select themed icons Displayed themed launcher icon in the home screen With this exercise it was possible to perform a manual configuration of the launcher icon for an Android application using the adaptive icon concept. Although it can also be done using SVG files, as recommended by Google's Android developers, in cases where you have png image files, this way can be very useful when creating these launcher icons. Adaptive icons not only apply to launchers, they can also be applied to notification icons and shortcut icons, but the content of this article covers launcher icons. ... Thank you for taking the time to read my article, it has been a while since I last wrote any long text in these parts. If you find something isn't quite right or have other information to add, feel free to add a respectful comment. If you like this article, please click the clap icon or share it on social media (or both). I Hope that you find this informative and useful and in some time you can use these steps in your android apps. Thanks for reading, Happy coding!! 😊 ... Use launcher adaptive icons, Official Android documentation tools to create icons for Android applications. I use this very simple tool Do you follow site's instructions or Erel's, in order to be compatible at b4a? 'This code will be applied to the manifest file during compilation. 'You do not need to modify it in most cases. 'See this link for for more information: AddManifestText( ) 'SetApplicationAttribute(android:icon, "@drawable/icon") SetApplicationAttribute(android:label, "$LABEL$") CreateResourceFromFile(Macro, Themes.LightTheme) 'End of default text. SetApplicationAttribute(android:icon, "@mipmap/ic_launcher") SetApplicationAttribute(android:roundIcon, "@mipmap/ic_launcher_round") CreateResource(mipmap-anydpi-v26, ic_launcher.xml,) No need to set any file to be read-only. Example: Hello After this method, do we have to put the main icon of the "==>Objects\res\drawable" and set file to be read-only. Or it is no longer necessary? thankyou does this also applies to the notification icons also? currently I use this code Dim n As Notification n.Initialize 'put file in res\drawable and set read-only n.Icon = "notification" can I move the notification.png file to the icon folder, with no read-only attribute? All app should have an adaptive icon. If you are not familiar with these icons then start here: Adaptive Icons Instruction steps: Create a new folder in the root project folder named icon. Create two folders inside that folder with the following files: mipmap: ic_launcher.png - non-adaptive icon for Android 7- devices. There is no specific size. Should be 128x128 or or more. background.png - 108x108 - the solid background layer. foreground.png - 108x108 - the foreground layer. mipmap-xxxhdpi (high resolution images): background.png - 432x432 - solid background layer foreground.png - 432x432 - foreground layer Add to main module: Add to manifest editor: SetApplicationAttribute(android:icon, "@mipmap/ic_launcher") CreateResource(mipmap-anydpi-v26, ic_launcher.xml,) No need to set any file to be read-only. Example: I missed this: And couldn't figure out where it was getting the icons Finally figured out it was using the ones in Objects\res\mipmap and Objects\res\mipmap-anydpi-v26 What do we do with these files? Page 2 I will wait for this resource to be added to Basic4Android because I tryed a lot but my program crash, so I ended up with simple "Choose Icon" on the menu that works fine. 1.- Download. 2. Unzip and find the android directory. 3. Add the resource directory where the icons are in your project. Note: rename android directory to icon 4. Add the directory where the icons are to your project. 4. Add the following to the manifest. SetApplicationAttribute(android:icon, "@mipmap/ic_launcher") SetApplicationAttribute(android:roundIcon, "@mipmap/ic_launcher_round") CreateResource(mipmap-anydpi-v26, ic_launcher.xml,) And here is the result, the icon which is not in the right dimension. Where is the problem please? Merci hello Unable to get the icon with correct size. I create an icon directory in which 2 subdirectories mipmap and mipmap-xxxhdpi. View attachment 152308 View attachment 152309View attachment 152310 I write in the Main: #SupportedOrientations: portrait #CanInstallToExternalStorage: False #AdditionalRes: ../icon And in the manifest : SetApplicationAttribute(android:icon, "@mipmap/ic_launcher") CreateResource(mipmap-anydpi-v26, ic_launcher.xml, Notice how the element is used to declare the and layers of the app icon by providing resource drawables for each. Go back to the Project view and locate the background and foreground drawables: res > drawable > ic_launcher_background.xml and res > drawable > ic_launcher_foreground.xml. Switch to Design view to see a preview of each: Background: Foreground: These are both vector drawable files. They don't have a fixed size in pixels. If you switch to Code view, you can see the XML declaration for the vector drawable using the element. ic_launcher_foreground.xml While a vector drawable and a bitmap image both describe a graphic, there are important differences. A bitmap image doesn't understand much about the image that it holds, except for the color information at each pixel. On the other hand, a vector graphic knows how to draw the shapes that define an image. These instructions are composed of a set of points, lines, and curves along with color information. The advantage is that a vector graphic can be scaled for any canvas size, for any screen density, without losing quality. A vector drawable is Android's implementation of vector graphics, intended to be flexible on mobile devices. You can define them in XML with these possible elements. Instead of providing versions of a bitmap asset for all density buckets, you only need to define the image once. Thus, reducing the size of your app and making it easier to maintain. Note: There are tradeoffs to using a vector drawable versus a bitmap image. For example, icons can be ideal as vector drawables because they are made up of simple shapes, while a photograph would be harder to describe as a series of shapes. It would be more efficient to use a bitmap asset in that case. Now it's time to move on to actually changing the app icon! Download the following two new assets that enable you to create an adaptive icon for the Affirmations app. You don't need to worry about understanding every detail of the vector drawable files. Their contents are auto-generated for you from design tools. Download ic_launcher_background.xml, which is the vector drawable for the background layer. If your browser shows the file instead of downloading it, select File > Save Page As... to save it to your computer. Download ic_launcher_foreground.xml, which is the vector drawable for the foreground layer. Note that there are certain requirements for these foreground and background layer assets, such as both must be 108 dpi x 108 dpi in size. You can view more details in the AdaptiveIconDrawable docs and you can also view design guidance on Android icons on the Material Design site. Because the edges of your icon could get clipped, depending on the shape of the mask from the device manufacturer, it's important to put the key information about your icon in the " safe zone." The safe zone is a circle of diameter 66 dpi in the center of the foreground layer. The content outside of the safe zone should not be essential, such as the background color, and okay if it gets clipped. Go back to Android Studio to use the new assets you just downloaded. First, delete the old drawable resources that contain the Android icon and green grid background. In the Project view, right-click on the file and choose Delete. Delete: drawable/ic_launcher_background.xml drawable/ic_launcher_foreground.xml Delete: mipmap-anydpi-v26/ mipmap-hdpi/ mipmap-mdpi/ mipmap-xhdpi/ mipmap-xxhdpi/ mipmap-xxxhdpi/ You can uncheck the box Safe delete (with usage search) and click OK. The Safe delete (with usage search) feature searches the code for usages of the resource you are about to delete. In this case, you will replace these folders with new ones of the same name, so you don't need to worry about Safe delete. Create a new Image Asset. You can either right-click on the res directory and choose New > Image Asset, or you can click on the Resource Manager tab, click the + icon, then select Image Asset from the dropdown. Android Studio's Image Asset Studio tool opens. Leave the default settings: Icon Type: Launcher Icons (Adaptive and Legacy) Name: ic_launcher With the Foreground Layer tab already selected, go to the Source Asset subsection. In the Path field, click the folder icon. A prompt pops up to browse your computer and select a file. Find the location of the new ic_launcher_foreground.xml file you just downloaded. It may be in the Downloads folder of your computer. Once you find it, click Open. The Path is now updated with the location of the new foreground vector drawable. Leave Layer Name as ic_launcher_foreground and Asset Type as Image. Next, switch to the Background Layer tab of the interface. Leave the default values. Click the folder icon in the Path field. Find the location of the ic_launcher_background.xml file you just downloaded. Click Open. The preview should update as you select the new resource files. This is what it should look like with the new foreground and background layers. By representing your app icon in two layers, device manufacturers—called original equipment manufacturers or OEMs for short—can create different shapes, depending on the Android device, as shown in the preview above. The OEM provides a mask that gets applied to all app icons on the device. When a circular mask is applied to both layers of your app icon, the result is a circular icon with an Android image and a blue grid background (left image above). Alternatively, a rounded square mask could be applied to produce the app icon in the above right. Having both a foreground and a background layer allows for interesting visual effects because the two layers can move independently of one another, and be scaled. For some fun examples of how the visual effects can look, view the Designing Adaptive Icons blogpost under Design Considerations. Because you don't know what device your user will have or what mask the OEM will apply to your icon, you need to set up your adaptive icon so important information doesn't get clipped. If important content is clipped or appears too small, then you can use the Resize slider bar under the Scaling section of each layer to make sure everything appears in the safe zone. To ensure nothing is clipped, resize the foreground and background images to 99% by dragging the Resize slider in the Foreground Layer and Background Layer tabs. Click Next. This step is to Confirm Icon Path. You can click the individual files to see the preview. Click Finish. Verify all the generated assets look correct in the mipmap folders. Examples: Great work! Now you'll make one more change. Test your app Test that your new app icon appears. Run the app on your device (emulator or physical device). Hit the Home button on your device. Swipe up to show the All Apps list. Look for the app you just updated. You should see the new app icon displayed. Note: Depending on your device model, you may see a launcher icon of a different shape. Nevertheless, it should show your foreground layer on top of your background layer with some type of mask applied to it. Nice job! The new app icon looks great. Adaptive and legacy launcher icons Now that your adaptive icon works well, you may wonder why you can't get rid of all the app icon bitmap images. You still need those files so that your app icon appears high-quality on older versions of Android, which is referred to as backwards compatibility. For devices running Android 8.0 or higher (API version 26 and above), Adaptive icons can be used (combination of foreground vector drawable, background vector drawable, with an OEM mask applied on top of it). These are the relevant files in your project: res/drawable/ic_launcher_background.xml res/drawable/ic_launcher_foreground.xml res/mipmap-anydpi-v26/ic_launcher.xml res/mipmap-anydpi-v26/ic_launcher_round.xml On devices running anything below Android 8.0 (but above the minimum required API level of your app), Legacy launcher icons are used (the bitmap images in the mipmap folders of different density buckets). These are the relevant files in your project: res/mipmap-mdpi/ic_launcher.webp res/mipmap-mdpi/ic_launcher_round.webp res/mipmap-hdpi/ic_launcher.webp res/mipmap-hdpi/ic_launcher_round.webp res/mipmap-xhdpi/ic_launcher.png res/mipmap-xhdpi/ic_launcher_round.webp res/mipmap-xxhdpi/ic_launcher.webp res/mipmap-xxhdpi/ic_launcher_round.webp res/mipmap-xxxhdpi/ic_launcher.webp res/mipmap-xxxhdpi/ic_launcher_round.webp Essentially, Android falls back to the bitmap images on older devices without adaptive icon support. Congratulations, you completed all the steps for changing an app icon! To download the code for the finished codelab, you can use these git commands: $ git clone $ cd basic-android-kotlin-compose-training-affirmations $ git checkout main Alternatively, you can download the repository as a zip file, unzip it, and open it in Android Studio. file_downloadDownload zip Note: The solution code is in the main branch of the downloaded repository. If you want to see the solution code, view it on GitHub. Navigate to the provided GitHub repository page for the project. Verify that the branch name matches the branch name specified in the codelab. For example, in the following screenshot the branch name is main. On the GitHub page for the project, click the Code button, which brings up a popup. In the popup, click the Download ZIP button to save the project to your computer. Wait for the download to complete. Locate the file on your computer (likely in the Downloads folder). Double-click the ZIP file to unpack it. This creates a new folder that contains the project files. Open the project in Android Studio Start Android Studio. In the Welcome to Android Studio window, click Open. Note: If Android Studio is already open, instead, select the File > Open menu option. In the file browser, navigate to where the unzipped project folder is located (likely in your Downloads folder). Double-click on that project folder. Wait for Android Studio to open the project. Click the Run button to build and run the app. Make sure it builds as expected. Place app icon files in the mipmap resource directories. Provide different versions of an app icon bitmap image in each density bucket (mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi) for backwards compatibility with older versions of Android. Add resource qualifiers onto resource directories to specify resources that should be used on devices with a certain configuration (v24 or v26). Vector drawables are Android's implementation of vector graphics. They are defined in XML as a set of points, lines, and curves, along with associated color information. Vector drawables can be scaled for any density without loss of quality. Adaptive icons were introduced to the Android platform in API 26. They are made up of a foreground and background layer that follow specific requirements, so that your app icon looks high-quality on a range of devices with different OEM masks. Use Image Asset Studio in Android Studio to create legacy and adaptive icons for your app. [[["Easy to understand","easyToUnderstand","thumb-up"],["Solved my problem","solvedMyProblem","thumb-up"],["Other","otherUp","thumb-up"]],[["Missing the information I need","missingTheInformationINeed","thumb-down"],["Too complicated / too many steps","tooComplicatedTooManySteps","thumb-down"],["Out of date","outOfDate","thumb-down"],["Samples / code issue","samplesCodeIssue","thumb-down"],["Other","otherDown","thumb-down"]],[],[],[]]