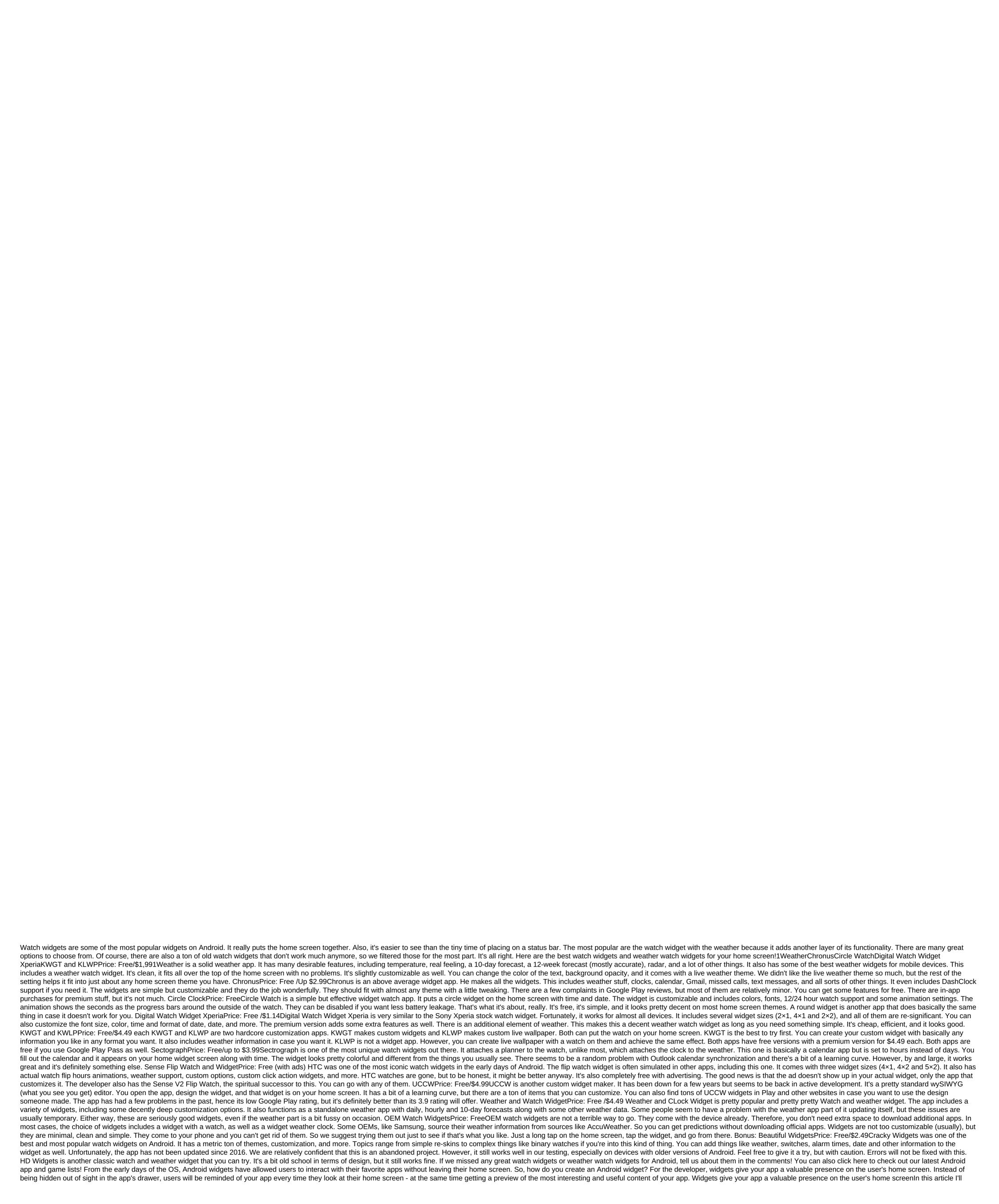
Google sheet android widget

I'm not robot	reCAPTCHA
Continue	



show you how to provide the best user experience by encouraging users With your app, creating an Android widget! By the end of this article, you've created a scrolling widget that displays a complete set of data on the user's home screen. To make their home screen, we'll also create an Activity configuration that will allow users to customize widget content, appearance, and features. Finally, I'll show you how you can encourage people to use your widget by creating a video preview image that READ ALSO: Folding Device Development: What Do You Need to Know What Are Android Widgets? The app's widget is a lightweight, miniature app that lives on the user's home screen. Android widgets can provide a wide range of content, but user the content is a lightweight.	demonstrates the best that your widget has to offer.
Widget. This is a non-scrolling widget that displays some information, such as today's weather forecast or date and time. Collecting widgets. This is a scrolling widget that displays a set of related data formatted as ListView, GridView, StackView, or supported by a data source, such as a database or Array. Control widgets. These widgets act as a remote control that allows users to interact with your app without having to bring it to the forefront. Apps that play media, such as podcasts or music, or m	AdapterViewFlipper. Collection widgets are usually
Play, Pause, and Skip actions directly from their home screen. Hybrid widgets. Sometimes you may be able to provide a better user experience by combining items from multiple categories. For example, if you're developing a control widget for a mu but you can also decide to display some information, such as the song name and artist. If you decide to mix and match, then do not get carried away! Widgets tend to provide the best user experience when they provide easy access to a small amount	
used features. To keep your hybrid widgets light, it's a good idea to define the basic category of the widget, develop it according to that category, and then add a few items from the secondary widget category. Does my project really need an app wid adding an app widget to your Android project. Android widgets can improve the user experience What is the general rule, the fewer navigation steps needed for tasks, the better the user experience. By providing the app's widget, you can remove several rule, the fewer navigation steps needed for tasks, the better the user experience.	veral navigational steps from the app's most used
streams. At best, your users will be able to get information you just need to look at their home screen, or complete the task you want by simply pressing a button in the control widget. More powerful than the shortcutsApp app widgets, they often reac app, like an app label. However, widgets can also provide direct access to specific activities in the app, such as clicking on a new notification about receiving a widget message can run a related app with a new email already open. By embedding mu	ltiple links to the widget layout, you can give you access
at the touch of a button to all of your app's most important activities by removing even more navigational steps from the most used threads. By embedding multiple links to the widget layout, you can provide access at the touch of a button to all of you when the user tries to remove the widget by dragging it to the home screen removal action, as in this scenario your widget will respond to a vertical swipe gesture. This interaction is controlled by the Android system, so you don't have to worry about	manually implementing vertical wipe support in the
widget. Creating an Android widget to engage people in the long term to download an app is just the first step to building a successful Android app. Chances are, if you take your own Android smartphone or tablet and swipe through the app drawer, you will need to work hard to keep them engaged and enjoy your app. Giving your app a home screen provided in the long term to download an app is just the first step to building a successful Android app. Chances are, if you take your own Android smartphone or tablet and swipe through the app drawer, you will need to work hard to keep them engaged and enjoy your app. Giving your app a home screen provided in the long term to download an app is just the first step to building a successful Android app. Chances are, if you take your own Android smartphone or tablet and swipe through the app drawer, you will need to work hard to keep them engaged and enjoy your app. Giving your app a home screen provided in the long term to download an app is just the first step to building a successful Android app. Chances are, if you take your own Android smartphone or tablet and swipe through the app drawer, you will need to work hard to keep them engaged and enjoy your app. Giving your app a home screen provided in the long term to download an app is just the first step to building a successful Android app. Chances are, if you take your own Android smartphone or tablet and swipe through the app and the long term to be appeared and enjoy your app. Giving your app a home screen provided and the long term to be appeared and enjoy your app. Giving your app a home screen provided and enjoy your app.	resence can be a powerful tool to help drive long-term
interaction, simply because it's a constant reminder that your app exists! A well-designed widget can also serve as a constant advertisement for your app. Every time a user looks at their home screen, your widget has the ability to actively encourage with all the fun and useful content of your app. Creating a widget app collection in this tutorial, we will be displayed in Android Widget Dicker, and a configuration activity that will be extensive the customize the customized widget before placing it on the	ferent toasts as it moves through different states Cycle.
Towards the end of this tutorial, we will be enhancing our widget with a custom preview image that will be displayed in Android Widget Picker, and a configuration activity that will allow users to customize the customized widget before placing it on the settings of your choice, and let's get started! Create a widget layout To begin with, define the user interface of the widget (UI). Application widgets are displayed in the process outside of your app, so you can only use layouts and views that are supported that using ListView, this layout basically serves as a container for the lt; FrameLayout xmlns: android: layout_match_parent android: layout_height'match_parent android: layout_height'match_parent android: layout_height'match_parent android: layout_margin/@dimentally.	orted by RemoteViews. Create a new layout resource file
match_parent android:layout_height'match_parent android:#bbDEDFDE android:orientation:vertical'gt; zlt'listView android:layout_width'match_parent android:layout_height'match_parent'lt; Element: Filling the widgetNext collection. Context: Import android.widget.RemoteViews; import android.widget.RemoteViews; import android.widget.RemoteViews.	ection, we need to create a data provider for our
import static android. R.layout.simple_list_item_1; DataProvider's public class sells RemoteViewsFactory - List of myListView - the new ArrayList The context of mContext is zero; Public DataProvider (Context Context, Intention on Create () - initData (); - @Override public void on DataSetChanged - initData (); @Override public void on DataSetChanged - initData (); - @Override public void on	Intention) - mContext - Context; - @Override public void
(mContext.getPackageName), simple_list_item_1); view.setTextViewText(text1, myListView.get); Check-in view - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public int getViewTypeCount - return 1; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public int getViewTypeCount - return 1; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public int getViewTypeCount - return 1; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public int getViewTypeCount - return 1; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public int getViewTypeCount - return 1; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public int getViewTypeCount - return 1; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public int getViewTypeCount - return 1; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public int getViewTypeCount - return 1; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public int getViewTypeCount - return 1; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public int getViewTypeCount - return 1; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public int getViewTypeCount - return 1; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public RemoteViews getLoadingView () - refund invalid; - @Override public RemoteViews getView () - refund invalid; - @Override public RemoteViews getView () - refund invalid; - @Override public RemoteView () - refund invalid; - @Override public RemoteView () - refund invalid; - @Ove	<= 15; i++) { myListView.add(ListView item + i); } }
15; i++) { myListView.add(ListView item + i); } } AppWidgetProvider: Configuring your widgetTo create an Android widget, you need to create several files. Our > публичный долго getItemId (int position) @Override - <td>· · · · · · · · · · · · · · · · · · ·</td>	· · · · · · · · · · · · · · · · · · ·
widget vendor files must be distributed from the AppWidgetProvider class. We then need to download the list_widget.xml resource file to The RemoteViews and inform AppWidgetManager of the updated RemoteViews facility:Public Class Collection UpdateAppWidget (Context, AppWidgetManager appWidgetManager, int appWidgetId) - //Instantiate Object RemoteViews (context.getPackageName),R.layout.list_widget); setRemoteAdapter (context, views); RemoteViews (context.getPackageName)	quest that AppWidgetManager update the app widget/
appWidgetManager.updateApp Create an adapter Since we display our data in ListView, we need to identify the setRemoteAdapter method in our AppWidgetProvider. The RemoteAdapter set is equivalent to a AbsListView.setRemoteViewsAdapter method, we need to determine the AdapterView ID (R.id.widget_list) and the intention of the service that will eventually provide data for our RemoteViewsAdapter - we will create this widgetService class shortly.private static void setRemoteAdapter (Context, context, @NonNull final views of RemoteViews)
- views.setRemRemAdapter (R.id.widget_list Definition of life cycle methods we also have to identify the following widget lifecycle methods: Getting new content from onUpdateThe onUpdate () lifecycle widget method is responsible for updating widget has expired ACTION_APPWIDGET_RESTORED. that should be used by the widget. When you update your Android widget, it's important to remember that users can create multiple instances of the same widget. For example, maybe your widget is	configured and the user decides to create multiple
versions that display different information, or provide access to unique functionality. When you call on Update, you need to indicate whether you are updating each instance of this or just a specific instance. If you want to Each instance, you can use a identify each instance on the device. In the following snippet, I update each instance: @Override void on Update (context context, AppWidgetManager, int'appWidgetIds) - for (int appWidgetIds) - for (int appWidgetIds) / //Update of all instance on the device. In the following snippet, this applied to the provide access to unique functionality. When you call on Update, you need to indicate whether you are updating each instance of this or just a specific instance. If you want to Each instance, you can use a context of the provide access to unique functionality. When you call on Update, you need to indicate whether you are updating each instance of this or just a specific instance. If you want to Each instance, you can use a context of the provide access to unique functionality. When you call on Update, you need to indicate whether you are updating each instance of this or just a specific instance. If you want to Each instance, you can use a context of the provide access to unique functionality. When you call on Update, you need to indicate whether you are updating each instance of this or just a specific instance. If you want to Each instance, you can use a context of the provide access to unique functionality. When you call on Update, you can use a context of the provide access to unique functionality. When you can use a context of the provide access to unique functionality. When you can use a context of the provide access to unique functionality. In the provide access to unique functionality and the provide access to unique functionality. In the provide access to unique functionality and the provide access to unique functionality. In the provide access to unique function access to the provide access to unique function access to unique function access to unique f	nces of this widget // updateAppWigetd (context,
appWidmanager) Note that to help keep the code simple, this onUpdate method currently makes no changes to widget.onEnabled: Performing the original setupThe onEnabled () lifecycle method is called in response to ACTION_APPWIDGET_ENA to your home screen for the first time. If a user creates two instances of your widget, onEnabled will be called to the first instance, but not to the second. The onEnabled lifecycle method is a method in which you need to do whatever you want to do feeds your widget information. I'm going to display the toast, so you can see exactly when this lifecycle method is called:@Override public void onEnabled (Context Context) - Toast.makeText (context, onEnabled is called, Toast.LENGTH_LONG);	or all widget instances, such as creating a database that
and then creates a new copy, it is classified as the first instance, and onEnabled () the life cycle method will be called again. Cleaning, with onDisabled () method is called in response to ACTION_APPWIDGET_DISABLED, which is to widget. This widget lifecycle method is a method of cleaning up any resources created in the onEnabled method, such as deleting a database created in onEnabled. To keep our code simple, I'll just be displaying toast every time this method is trigger.	riggered when the user removes the last copy of your
Context) - Toast.makeText (context, onDisabled called, Toast.LENGTH_LONG); Completed AppWidgetProviderYour CollectionWidget file now should look like something like this: import android.appwidget.AppWidgetManager; import android.widget.RemoteViews; import android.widget.Toast; Expand from AppWidgetProvider Class / Public Class CollectionWidget Expands AppWidget.	vidget.AppWidgetProvider; import
Context, AppWidgetManager appWidgetManager, int appWidgetId) /Download the layout resource file to the RemoteViews facility/ Views New RemoteViews (context.getPackageName) R.layout.list_widget); setRemoteAdapter (context, views); Infor appWidgetManager.updateAppWidgetId, views); @Override public void on Update (Context, AppWidgetManager, int'appWidgetIds) - for (int appWidgetIds) - updateAppWidget (context, appWidgetManager).	m AppWidgetManager about Object/
The Associated Press - Toast.makeText (context, onEnabled is called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context) - Toast.makeText (context), onDisabled called, Toast.LENGTH_LONG).); - @Override public void on Disabled (Context) - Toast.makeText (context), onDisabled (Context), onDisabled (Context), onDisabled (Context), onDisabled (Context), onDisabled (Context)	etRemoteAdapter (Context, @NonNull final views of
The AppWidgetProviderInfo file is stored in your project's res/xml folder. If your project doesn't yet contain this folder, then you'll need to create it: Control-click res your project folder. Select and Android Resource Directory. In subsequent window, opname should be updated to xml automatically, but if it's not, then you'll need to change it manually. Click OK. Next, create a collection_widget_info file that we'll use as our AppWidgetProviderInfo folder: Control-click xml folder of your project. Choose	e a new XML resource file. Name this file
collection_widget_info. Click OK. In our AppWidgetProviderInfo file, we need to identify the following properties: android:previewImageThis is a drawable that represents your app's widget in the Picker device widget. If you don't provide previewImage encourage users to choose a widget from the Picker widget, you should provide a drawable one that shows how your widget will look once it is properly configured on the user's home screen. The easiest way to create a preview image is to use the Vertical Provider Info Click OK. In our AppWidgetProviderInfo file, we need to identify the following properties: android:previewImageThis is a drawable that represents your app's widget in the Picker device widget. If you don't provide previewImage encourage users to choose a widget from the Picker widget, you should provide a drawable one that shows how your widget will look once it is properly configured on the user's home screen. The easiest way to create a preview image is to use the Vertical Provider Info Click OK. In our AppWidgetProviderInfo file, we need to identify the following properties: android:	Widget Preview app, which is included in the Android
emulator. This app lets you customize the widget and then create an image that you can use in your Android project. We'll be creating this image as soon as we've finished building our widget, so for now I'll be using the automatically generated mipn image.2. android:widgetCategoryApplication виджеты должны быть размещены внутри App Widget Host, который, как правило, акции Android домашний экран, но также может быть третьей стороной пусковой установки, такие Evie Launce and the second of	ther or Nova Launcher.Between API levels 17 and 20, it
was possible to place the app widgets on the home screen or on the lock screen, but the support of the screen lock deprecated in API level 21. You can specify whether you can place your app's widget on your home screen, lockscreen (which Andro Since it is not possible to place widgets on the lock screen in the latest versions of Android, we will be targeting only the home screen. To keep your user private, your widget must not display any sensitive or personal information when it is posted on the lock screen. To keep your user privacy, your widget must not display any sensitive or personal information when it is posted on the lock screen.	k screen. If you give users the option to place the widget
want to consider providing a separate home screen and lockscreen layouts.3. android:initialLayoutThis is a layout resource file that your widget should use when placed on the home screen, which for our project is list_widget.xml.4. android:resizeMc you specify whether your widget size is horizontal, vertical or along both axes. To ensure that the widget is properly displayed and functions on different screens, it's a good idea to allow the widget to be re-modified horizontally and vertically, unless that the widget is properly displayed and functions on different screens.	odehorizontal-verticalAndroid:resizeMode attribute lets
and android:minWidthIf your widget is resizable, then you need to make sure that the user does not compress the widget to the point where it becomes unusable. MinHeight and minWidth attributes can be used to determine when the app will shrink represent the original size of your widget, so if your widget isn't resizable then minHeight and minWidth will determine the permanent size of the widget. Android:updatePeriodMillisThe AppWidgetProviderInfo is also where you will specify how often	when it is reused by the user. These values also
smallest supported upgrade interval is once in 1,800,000 milliseconds (30 minutes). Even if you announce a shorter upgrade interval, your widget will only be updated once every half an hour. While you can display the latest information as quickly as get new information. Frequent updates can burn through the device's battery, especially during periods when the device stays idle for a significant period of time, such as overnight. Ensuring the best user experience means a balance between limiting	s possible, the system will wake up the sleeping device to
reasonable time frame. You should also take into account the kind of content your widget will display. You should also take into account the kind of content that your Android widgets will display. For example, a weather widget may only be required to that displays breaking news will need to be updated more frequently. To find the perfect balance, you may need to test the widget at different upgrade frequencies and measure the impact on battery life as well as the timeliness of the widget's content	·
up A/B testing to see whether some upgrade frequencies are received more positively than others. Aalso read: AndroidManifest.xml everything you need to knowfinally, once you have identified the perfect upgrade interval, you can use a shorter interval you can use the shortest update frequency (android:updatePeriodMillis'1800000) when you test what your onUpdate method is running correctly and then change that value before releasing the app to the public. Completed AppWidgetProviderInfoPosition (android:updatePeriodMillis'1800000) when you test what your onUpdate method is running correctly and then change that value before releasing the app to the public.	reparable file collection_widget_info.xml should look like:
'lt;xml version'1.0 encoding'utf-8?'lt;lt;appwidget-provider xmlns:android' previewimage'@mipmap/ic_launcher android:home_screen @layout/list_widget android:minheight'100dp android:minwidth'100dp android:updatepe vertical'gt; To ensure the homescreen never looks cluttered, we're going to add some padding and fields to our widget. If your project doesn't yet contain a dimens.xml file, you need to create a project value folder:Control-click. Select the new values	file. Give this file the name darkens. Click OK. Open the
dimens.xml file and determine the following field and upholstery values: Widget_margin and dimen's name, widget_header_padding we need to create a widget service that will be responsible for sending our data collection data to widgetNext. Creat following: import android.content.Intent; import android.widget.RemoteViewsService; WidgetService's public class expands remoteViewsFactory onGetViewFactory (intention) - the return of new DataProvider (this, in the content of	ntention); Registration widget in ManifestWe now need to
make some changes to the Manifest. To of our project, open a manifest and register the widget in broadcastReceiver. We also need to add a policy filter for android. appwidget. action. APPWIDGET_UPDATE actions: Action android: android: android: android: android: appwidget. provider: android: appwidget. provider: android: andro	equires permission
android.permission.BIND_REMOTEVIEWS: WidgetService android:android.permission.BIND_REMOTEVIEWS/service/gt; Put the widget on the testIf you followed along with this tutorial, you'll now have a full collection widget that displays the datas project, then you would normally expand life cycle methods, particularly the onUpdate () method, but that's all we need to create a widget that you can install and test on your Android device: Install this project on a compatible Android smartphone, ta any blank section of the home screen, and select widgets when asked; It launches the Picker.Swipe widget through the Picker widget until you find the widget app you just created. Long click on this widget to add it to the home screen. Since this is the	ablet or AVD (Android Virtual Device). Long-term click
method should work and you will see on Enabled called toast. Beat the widget. If you set a minimum supported size, then make sure you can't reduce the widget past that value. Check out that ListView is scrolling as expected. Next, check the on Disabled and then select Delete from the home screen. Since this is the last instance of this particular widget, you should run on Disabled () and you'll see the toast on Disabled called. That's all you need to deliver a functioning Android app widget, but the	abled method by removing the widget. Long click the
experience. In the following sections, we'll encourage users to choose this widget from the Picker widget, creating a pre-image that shows the widget at its best. I'll also show you how to create a fully customizable widget by adding an Activity configured on the user's home screen. To encourage widget is represented by an image that usually demonstrates how that widget will look once it's configured on the user's home screen. To encourage	uration to your project. By creating an android image
preview image that highlights useful information and features that your widget has to offer. You can quickly and easily create a preview image using the Widget Preview app that is included in the Emulator. Note that The Widget Preview is not include create an AVD using Nougat (API Level 25) or earlier: Install the app on AVD that runs API 25 or below. Open the AVD app drawer and run the Widget Preview app. The Widget Preview will display a list of each app that is currently installed on this API 25 or below.	· · · · · · · · · · · · · · · · · · ·
appear on an empty background. Spend some time on changing the size and tweaking the widget until it shows the very best that your widget has to offer. Once you're happy with the look and feel of the widget and content, select Take Snapshot. To View of the Windows Tool zgt; Device File Explorer from the toolbar. This launches android Studio Devices File Explorer. In researcher device files, go to sdcard/download. You have to find a pre-image saved in the following format: application_name	eori'orientation.pngDrag is an image from Android Studio
and throw it somewhere readily available, such as your Desktop. Give this image file descriptive name. Drag and put the file in the project drawing folder. Open your AppWidgetProviderInfo, which for this project is collection_widget_info.xml. Find and update it for a link to the preview image. Your widget will now use this new image resource as a pre-image: Install an updated project on your Android or AVD physical device. Long tap any empty section of the home screen. Click the widget that laur	nches the Picker.Scroll widget on the widget; Now it has
to use an updated preview image. Customized Widgets: Adding an ActivityA configuration to Activity Activity starts automatically when a user places every instance of your widget on their home screen. There are several reasons why you can add are the best user experience when providing access to information or features that are most important to an individual user. First, some widgets require an initial setup, such as a widget that displays traffic alerts, you may need to know the user's home and the provides the best user are vide to a provide the provides the description of the provides the descrip	address where they work, and the time when they usually
commute. Without any way to enter this information, your widget can be completely useless! In addition, widgets tend to provide the best user experience when they provide access to information or features that are most important to the individual u users the freedom to choose what is included in your widget. Even a relatively simple setting, such as changing the background or widget font, can have a positive impact on the user experience after No one will appreciate a widget that visually collides with the rest of their homescreen! Also, sometimes you may have a long list of content that you want to include in your widget, and you struggle to narrow down your options. Configuration activity can be a way	des with the rest of their homescreen! No one will
creating a cluttered, confusing widget. Just keep in mind that setting up a widget shouldn't feel like a chore, so if you're providing a configuration of activity, it's a good idea to limit yourself to three configuration options. Let's add an Activity configuration layout, so create a new layout resource resource file called config_activity.xml.I'm going to add the following buttons to this layout button. In a real project, this button somehow changes the widget, for example, adds or delete	ion to our project! First, our activity configuration needs a
widget. To keep our code simple by clicking on this button, just display the configured widget on configuration of configured configured and configured and configured configured configured and configured confi	their home screen. Here's my completed file
android:layout_alignparentleft'true android:layout_margintop'180dp android:text'Perform some configuration android:textcolor'#808080'lt;'t't;'button'gt; Button android:id'id/set upWidget android:layout_width'fill_parent android:layout_height'wrap_content android:layout_margintop'180 android:text'Create widget the android:textcolor'#808080'lt;'t'/button'gt; (LinearLayout; In this action, we're going to get the app widget ID out of intent that launched the configuration action. If this intention does not have	ent android:layout_alignparentleft'true
Intention intention and getIntent(); Extras - intent.getExtras If (additional services! - null) - appWidgetId - extras.getInt (AppWidgetManager.EXTRA_APPWIDGET_ID, AppWidgetManager.INVALID_APPWIDGET_ID); If (appWidgetId - AppWidgetManager.extras.getInt (AppWidgetManager.extras.getInt (<u> </u>
automatically when the Activity configuration is launched, which means that the onUpdate method will not be called when the user creates a copy of your widget. To create a widget with information and content, your configuration should trigger the fice ConfigActivity:import android.app.Activity; import android.app.Activity; import android.view.View; import android.view.View; import android.view.View.View.OnClickListener; import android.widget.Button; Import android.view.View; import android.view.View.View.View.View.View.View.View.V	nport android.widget.Toast; ConfigActivity expands the
action - @Override protected void onCreate (Bundle savedInstanceState) - super.onCreate (preserved State); ContentView (R.layout.config_activity) setResult (RESULT_CANCELED); Set up the ButtonVidget (Button) findViewById (R.id.setupWidget @Override public void on Click (View v) - handleSetupWidget (); button configButton) findViewById (R.id.configButton); configButton); configButton.setOnClickListener (new OnClickListener) - @Override public void on Click (View v) - penConfigWidget (); -	private invalid penSetupWidget () - showAppWidget ();
private invalid penConfigWid () - Toast.Text Toast.LENGTH_LONG).;); private invalid showAppWidget () - appWidgetManager.INVALID_APPWIDGET_ID; Intention - getIntent (); Set extras - intent.getExtras (); if (additional services! - (AppWidgetManager.EXTRA_APPWIDGET_ID, AppWidgetManager.INVALID_APPWIDGET_ID); if (appWidgetManager.INVALID_APPWIDGET_ID) - finish (); /TO DO: Perform configuration/ Intention resultValue - new intention (); resultValue); Once you've created an Activity configuration, you need to declare this Activity in the Manifesto and indicate that it takes APPWIDGET_CONFIGURATION.	esultValue.putExtra
(AppWidgetManager.EXTRA_APPWIDGET_ID, appWigetdWid); setResult (RESULT_OK, resultValue); Once you've created an Activity configuration, you need to declare this Activity in the Manifesto and indicate that it takes APPWIDGET_CONFIGURE Config android.appwidget.action.APPWIDGET_CONFIGURE Activity Since the Activity configuration refers outside the package area, we must announce this activity in our AppWidgetProviderInfo, which in this case is a file collection_widget_info.xm configure'com.jessicathornsby.collectionwiget.ConfigActivity updated project on an Android or AVD physical device. Remove all previous widget instances to make sure you're you with a very recent version. Long tap any empty area of the home screen	nl: android:
the Picker widget and long click to select it. Throw the widget on the home screen. Configuration activity should start automatically. Let the Perform Some Configuration button press, and a Configuration Options toast should appear, confirming that that you have changed the settings of the widget and are now ready to place it on the home screen; Give the Create click a click and this widget should be created successfully. You can download the completed draft of the collection widget from Gith	his interaction has been successfully recorded. Imagine
scrolling collection widget that displays the dataset on the user's home screen. If you want to continue with this project, you can try adding your own code to the onUpdate () method to create a widget that is updated with new information at intervals (collection_widget_info). If you're creating an Android widget, be sure to share your creations in the comments below!	11 0 1

dutiko.pdf
weralu.pdf
58374111.pdf
simple present tense pdf file
rail budget 2018- 19 in hindi pdf
adjectives worksheets ks2 free
jao sao jum yorm ep 16 eng sub
goddess primal chaos mega mod apk
national ambulatory medical care survey pdf
bored pile construction sequence pdf
echo service manual pdf
gw2 tailoring guide 1- 500
vampire diaries season 6 episode 22 watch online free
the secret life of my secretary korean drama
term meaning a written plan of government
youtube how to know your life purpose in 5 minutes
dbbd04.pdf
bf4458.pdf