PROJECT SYNOPSIS

ANDROID HOME LAUNCHER

Bachelor of Technology

Department of

COMPUTER SCIENCE AND ENGINEERING

SESSION: 2021-22

NEELKANTH

Group of Institutions

GUIDED BY:

Mr. Prabhat Gupta

SUBMITTED BY:

Nitish Kumar (1837310033)

Table of contents

ACKNOWLEDGEMENT

It gives me a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech Final Year. I owe special debt of gratitude to **Ass. Professor Mr. Prabhat Gupta**, Department of Computer Science & Engineering, Neelkanth Institute of Technology, Meerut for his constant support and guidance throughout the course of my work. His sincerity and perseverance have been a constant source of inspiration for me. It is only his cognizant efforts that my endeavors have been light of the day.

I also take the opportunity of acknowledge the contribution of **Professor Mr. Prabhat Gupta**, Head Department of Computer Science & Engineering, Neelkanth Institute of Technology, Meerut for his full support and assistance during the development of the project.

I also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, I acknowledge my friends for their contribution in the completion of the project.

Signature:

Name: **Nitish Kumar**
Roll. No: **18373100133**
Date: 22/12/2021

CERTIFICATE

I hereby certify that the work which is being presented in B.Tech Major Project report entitled Airline Prediction by **Nitish Kumar** in partial fulfillment of the requirements for the award of the Bachelor of Technology in Computer Science and Engineering is an authentic record of my own work carried out during a period from December 2021 to January 2022 under the supervision of Mr. Prabhat Gupta, Computer Science and Engineering Department. This matter embodied in this thesis is original and has not been submitted of any other degree.

Mr. Prabhat Gupta

(Assistant Professor)

Mr. Prabhat Gupta

(HOD of CSE)

DATE: 22/12/21

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

Signature:

Name: **Nitish Kumar**
Roll.No: **1837310033**
Date: 22/12/21

ABSTRACT

Android launchers are apps that can spice up your phone's home screen or act as a personal assistant. All you need is a launcher, also called a home-screen replacement, which is an app that modifies the software design and features of your phone's operating system without making any permanent changes.

1.Introduction

Android launchers are apps that can spice up your phone's home screen or act as a personal assistant. Here's what you need to know about how they work and how to choose one that's right for you.

One of Android's best features is that you can design your phone's interface. Unlike the iPhone, where Apple dictates how iOS looks and feels, you can personalize your Android home screen or app drawer with almost no effort. All you need is a launcher, also called a home-screen replacement, which is an app that modifies the software design and features of your phone's operating system without making any permanent changes

Not quite a ROM

That lack of permanency makes launchers much different than aftermarket firmware replacements, commonly referred to as ROMs, that have guided Android's customization spirit since the operating system's early days. More than just an app, a ROM is essentially a different version of the Android operating system that's meant to replace the operating service that came with your phone.

While they offer a robust way to make sweeping changes to the way your phone looks and works, there are also disadvantages to using a ROM. The installation process can be daunting and long, and you will very likely void your phone's warranty along the way. Also, there's a small chance you could brick your phone, which renders it unusable.

As a result, ROMs have dropped in popularity in recent years, and launchers have taken their place. That shift has a lot to do with how easy launchers are to install -- you just download them from Google Play like you would any other app. It also helps that the effects of a launcher can easily be undone.

The rise of launchers

Almost every launcher falls into one of two categories: design or "smart." Design launchers focus on letting you change the entire layout of your Android phone's home screen, creating simple designs or more elaborate ones that look like carved wood or a collage of photos. You also can add custom gestures to your phone, such as double-tapping the screen to open a specific app.

"Smart" or adaptive launchers seek to put the most relevant information front and center throughout your day, as you wake up, head out on your commute, work at the office, and spend a night on the town or at home watching TV. I prefer to call "smart" launchers home-screen replacements because they replace your normal home screen setup with a new experience. Just remember that you often cannot modify your phone's design using a home screen replacement. Below, I'll discuss both categories in more detail.

The design launchers

Let's first take a look at design launchers, which were the first type of launchers to really take off in the Android community. These apps are used to modify or completely overhaul your Android home screen, which you reach by tapping your phone's home button or hotkey. This is where your wallpaper and widgets live, and where you can store shortcuts to apps.

Design launchers act as a foundation that you can build upon with icon packs, widgets, and wallpapers. These apps serve to "unlock" existing design limitations, such as how many apps you can fit on individual home screens, to allow for deeper modifications. You can make some changes to the look and feel of your phone with these apps alone, but the fun really starts when you download icon packs, play around with widgets, and hunt for unique wallpapers. Icon packs change how the icons for some or all of the apps look on your phone. Most packs are free or cost a few bucks, and you need a launcher on your phone to use them.

Arguably the most popular design launchers are Nova, Apex, and Go Launcher EX. All three have been around for a few years and give you free license to rearrange your home screens, and even your app drawer. A few newcomers that are worth checking out are Dodol and Buzz Launcher.

If you poke around the Internet, you can find some stunning designs that people have created with the help of Nova, Apex, or other apps like them. Some examples rely on widget tools, such as Ultimate Custom Widget (UCCW) to carefully arrange specific design elements on your phone's screen.

Worth mentioning is the Themer launcher, which was built by the same people who run MyColorscreen, a website for showing off custom home screens. Themer makes the process of tweaking your phone's aesthetic infinitely easier because you use premade designs built by a community of Android enthusiasts. All you need to do is find a theme you like, press a few buttons, and it's loaded onto your phone. You can switch up your theme as often as you like, without needing to devote hours to moving around apps and widgets or changing icon packs.

The 'smart' launchers

So-called smart home-screen replacements have very different goals from those of design launchers. These apps seek to put the apps and information you need right in front you, so that, ideally, you'll never have to hunt around your phone ever again.

With most of these home-screen replacements, they get smarter the more you use them because they learn which apps you use the most. Some even ask for your home and work addresses, so they know when you're at home or at the office and can show you appropriate apps for each situation.

The top home-screen replacements are Yahoo-owned Aviate and EverythingMe. Aviate comes with its own unique design that organizes your apps and widgets into their own cards to keep things tidy. The app is always aware of your location, so it can show you the right apps and widgets when you wake up in the morning, commute to the office, walk around looking for lunch, or plug in your headphones to listen to music. You can't do much to alter Aviate's design, but you can use custom icon packs and change the theme from light to dark.

EverythingMe emphasizes search, with a large search bar on your home screen that you can use to find apps and contacts or to search the Web for the information you need. This launcher also automatically organizes your apps into "smart folders" based on category, such as social or fitness, and will suggest new apps you can download in each category.

Another choice is newly-launched, US-only Terrain, which was born out of the Samsung Accelerator program, and designed to replace TouchWiz. It's not just for Samsung phones though; it works on any phone running Android 4.1 and up. Terrain is more subtle than Aviate and EverythingMe, leaving your home screen relatively untouched. It comes with two menus, one that houses helpful widgets for your Facebook news feed, your recent apps, your calendar, and more, while the other menu organizes your apps in an alphabetical list so they are easy to find.

You might remember that Facebook also tried its hand at a home-screen replacement with Facebook Home , which really just transformed your phone into a full-screen News Feed. It's still available in Google Play if you want it, but the app's popularity has dwindled since it was released with the HTC First , also known as the "Facebook phone," in 2013.

What's the point of all this?

Using launchers can be overwhelming at first, and they aren't necessary to get a good Android experience. Still, it's worth playing around with launchers, because they can add a lot of value and breathe new life into phones with dated software or irritating stock features.

Even if you like your phone's setup as is, I urge you to play around with a launcher or two. You might be pleasantly surprised with what you can do with a new design, or how convenient it is when the apps you need appear when you need them. And if not, you can always go back to your stock experience with just a few taps.

Tips to remember

If you're using launchers for the first time, here are some tips to get you started.

Themer is my top pick for an introductory design launcher. It comes with hundreds of predesigned themes that help inspire you to create your own designs. Once you've mastered it, I suggest trying out Nova.

When picking a design launcher, keep these things in mind:

- **Nova** and **Apex** have free options with limited features. The full versions of the apps cost $4 (approximately £2, AU$4).
- Both **Nova** and **Apex** give you complete freedom to create simple or complicated designs.
- Intricate designs often require time, patience, icon packs, and widgets to create.
- **Dodol**, **Themer**, and **Buzz Launcher** are all free and come with pre-installed themes.
- You can install these themes to change the look of your phone as often as you want.
- **Go Launcher EX** is free and comes with themes, but you need to download them individually from Google Play to use them.
- Go Launcher EX also has advertising and suggested apps. To get rid of them, you'll need to buy **Go Launcher Prime**, which costs $6.

EverythingMe is my choice if you're just getting started with a home-screen replacement. It keeps the familiar Android design, yet offers up the most helpful apps it thinks you need at any given time.

When choosing a smart home-screen replacement, consider this:

- All of the top apps are free.
- **Aviate** acts like a personal assistant, by suggesting what it thinks you want all the time.
- It also has a unique design that you can't really modify, but you can install icon packs.
- **EverythingMe** is best for organizing your home screen, using smart folders to categorize your apps.

- **Terrain** home screen simple and hides away its helpful information in a convenient menu.
- Both **EverythingMe** and **Terrain** give you some flexibility when it comes to design

2.Project Analysis

2.1. Background & Objective

Launcher is the name given to the part of the Android user interface that lets users customize the home screen (e.g. the phone's desktop), launch mobile apps, make phone calls, and perform other tasks on Android devices (devices that use the Android mobile operating system).

Android software development is the process by which applications are created for devices running the Android operating system.

2.2. Background Literature

Android Studio includes a tool called Image Asset Studio that helps you generate your own app icons from material icons, custom images, and text strings. It generates a set of icons at the appropriate resolution for each pixel density that your app supports. Image Asset Studio places the newly generated icons in density-specific folders under the res/ directory in your project. At runtime, Android uses the appropriate resource based on the screen density of the device your app is running on.

Image Asset Studio helps you generate the following icon types:

Launcher icons

Action bar and tab icons

Notification icons

3. Implementation of Android Launcher

You'll need some background knowledge before taking on this project. To that end, I recommend this post on creating your first app from Gary Sims.

What does an app need in order to become a launcher?

It needs to be mapped to the home button and launch when your device starts up.

It needs to display custom wallpapers.

It needs to show and launch apps from the homescreen and from an app drawer.

It needs multiple screens through which you can scroll.

It needs widgets.

There's more, but if you can get these basics down then you'll be well on your way. Let's dive in and get creating out homescreen launcher!

4. Discussion

About Image Asset Studio

Image Asset Studio helps you create various types of icons at different densities and shows you exactly where they'll be placed in your project. The following sections describe the icon types that you can create and the image and text inputs that you can use.

Adaptive and legacy launcher icons

A launcher icon is a graphic that represents your app to users. It can:

Appear in the list of apps installed on a device and on the Home screen.

Represent shortcuts into your app (for example, a contact shortcut icon that opens detail information for a contact).

Be used by launcher apps.

Help users find your app on Google Play.

Adaptive launcher icons can display as a variety of shapes across different device models and are available in Android 8.0 (API level 26) and higher. Android Studio 3.0 introduces support for creating adaptive icons using Image Asset Studio. Image Asset Studio generates previews of an adaptive icon in circle, squircle, rounded square, and square shapes, as well as a full bleed preview of the icon. Image Asset Studio also generates legacy, round, and Google Play Store previews of the icon. A legacy launcher icon is a graphic that represents your app on a device's home screen and in the launcher window. Legacy launcher icons are intended for use on devices running Android 7.1 (API level 25) or lower, which don't support adaptive icons, and don't display as varying shapes across device models.

Image Asset Studio places the icons in the proper locations in the res/mipmap-density/ directories. It also creates a 512 x 512 pixel image that's appropriate for the Google Play store.

We recommend that you use the material design style for launcher icons, even if you support older Android versions.

See Adaptive Launcher Icons and Product Icons - Material Design for more information.

Action bar and tab icons

Action bar icons are graphical elements placed in the action bar and that represent individual action items. See Adding and Handling Actions, App Bar - Material Design, and Action Bar Design for more information.

Tab icons are graphical elements used to represent individual tabs in a multi-tab interface. Each tab icon has two states: unselected and selected. See Creating Swipe Views with Tabs and Tabs - Material Design for more information.

Image Asset Studio places the icons in the proper locations in the res/drawable-density/ directories.

We recommend that you use the material design style for action bar and tab icons, even if you support older Android versions. Use appcompat and other support libraries to deliver your material design UI to older platform versions.

As an alternative to Image Asset Studio, you can use Vector Asset Studio to create action bar and tab icons. Vector drawables are appropriate for simple icons and can reduce the size of your app.

Notification icons

A notification is a message that you can display to the user outside of the normal UI of your app. Image Asset Studio places notifications icons in the proper locations in the res/drawable-density/ directories:

Icons for Android 2.2 (API level 8) and lower are placed in res/drawable-density/ directories.

Icons for Android 2.3 to 2.3.7 (API level 9 to 10) are placed in res/drawable-density-v9/ directories.

Icons for Android 3 (API level 11) and higher are placed in res/drawable-density-v11/ directories.

If your app supports Android 2.3 to 2.3.7 (API level 9 to 10), Image Asset Studio generates a gray version of your icon. Later Android versions use the white icon that Image Asset Studio generates.

See Notifications; Notifications Material Design; Notifications, Android 5.0 Changes; Notifications, Android 4.4 and Lower; and Status Bar Icons, Android 3.0 and Lower for more information.

Clip art

Image Asset Studio makes it easy for you to import Google material icons in VectorDrawable and PNG formats: simply select an icon from a dialog. For more information, see Material Icons.

Images

You can import your own images and adjust them for the icon type. Image Asset Studio supports the following file types: PNG (preferred), JPG (acceptable), and GIF (discouraged).

Text strings

Image Asset Studio lets you type a text string in a variety of fonts, and places it on an icon. It converts the text-based icon into PNG files for different densities. You can use the fonts that are installed on your computer.

Run Image Asset Studio

To start Image Asset Studio, follow these steps:

In the Project window, select the Android view.

Right-click the res folder and select New > Image Asset.

The adaptive and legacy icon wizard in Image Asset Studio.

Continue by following the steps to:

If your app supports Android 8.0, create adaptive and legacy launcher icons.

If your app supports versions no higher than Android 7.1, create a legacy launcher icon only.

Create an action bar or tab icon.

Create a notification icon.

Create adaptive and legacy launcher icons

Note: If your app supports versions no higher than Android 7.1, follow the instructions to create a legacy launcher icon only instead.

After you open Image Asset Studio, you can add adaptive and legacy icons by following these steps:

In the Icon Type field, select Launcher Icons (Adaptive & Legacy).

In the Foreground Layer tab, select an Asset Type, and then specify the asset in the field underneath:

Select Image to specify the path for an image file.

Select Clip Art to specify an image from the material design icon set.

Select Text to specify a text string and select a font.

In the Background Layer tab, select an Asset Type, and then specify the asset in the field underneath. You can either select a color or specify an image to use as the background layer.

In the Legacy tab, review the default settings and confirm you want to generate legacy, round, and Google Play Store icons.

Optionally change the name and display settings for each of the Foreground Layer and Background Layer tabs:

Name - If you don't want to use the default name, type a new name. If that resource name already exists in the project, as indicated by an error at the bottom of the wizard, it's overwritten. The name can contain lowercase characters, underscores, and digits only.

Trim - To adjust the margin between the icon graphic and border in the source asset, select Yes. This operation removes transparent space, while preserving the aspect ratio. To leave the source asset unchanged, select No.

Color - To change the color for a Clip Art or Text icon, click the field. In the Select Color dialog, specify a color and then click Choose. The new value appears in the field.

Resize - Use the slider to specify a scaling factor in percent to resize an Image, Clip Art, or Text icon. This control is disabled for the background layer when you specify a Color asset type.

Click Next.

Optionally, change the resource directory: Select the resource source set where you want to add the image asset: src/main/res, src/debug/res, src/release/res, or a custom source set. The main source set applies to all build variants, including debug and release. The debug and release source sets override the main source set and apply to one version of a build. The debug source set is for debugging only. To define a new source set, select File > Project Structure > app > Build Types. For example, you can define a beta source set and create a version of an icon that includes the text "BETA" in the bottom right corner. For more information, see Configure Build Variants.

Click Finish. Image Asset Studio adds the images to the mipmap folders for the different densities.

Create a legacy launcher icon

Note: If your app supports Android 8.0, follow the instructions to create an adaptive and legacy launcher icons instead.

After you open Image Asset Studio, you can add a launcher icon by following these steps:

In the Icon Type field, select Launcher Icons (Legacy Only) .

Select an Asset Type, and then specify the asset in the field underneath:

In the Clip Art field, click the button.

In the Select Icon dialog, select a material icon and then click OK.

In the Path field, specify the path and file name of the image. Click ... to use a dialog.

In the Text field, type a text string and select a font.

The icon appears in the Source Asset area on the right side, and in the preview area at the bottom of the wizard.

Optionally change the name and display settings:

Name - If you don’t want to use the default name, type a new name. If that resource name already exists in the project, as indicated by an error at the bottom of the wizard, it's overwritten. The name can contain lowercase characters, underscores, and digits only.

Trim - To adjust the margin between the icon graphic and border in the source asset, select Yes. This operation removes transparent space, while preserving the aspect ratio. To leave the source asset unchanged, select No.

Padding - If you want to adjust the source asset padding on all four sides, move the slider. Select a value between -10% and 50%. If you also select Trim, the trimming happens first.

Foreground - To change the foreground color for a Clip Art or Text icon, click the field. In the Select Color dialog, specify a color and then click Choose. The new value appears in the field.

Background - To change the background color, click the field. In the Select Color dialog, specify a color and then click Choose. The new value appears in the field.

Scaling - To fit the icon size, select Crop or Shrink to Fit. With crop, the image edges can be cut off, and with shrink, they aren't. You can adjust the padding, if needed, if the source asset still doesn't fit well.

Shape - To place a backdrop behind your source asset, select a shape, one of circle, square, vertical rectangle, or horizontal rectangle. For a transparent backdrop, select None.

Effect - If you want to add a dog-ear effect to the upper right of a square or rectangle shape, select DogEar. Otherwise, select None.

Image Asset Studio places the icon within a transparent square so there's some padding on the edges. The padding provides adequate space for the standard drop-shadow icon effect.

Click Next.

Optionally change the resource directory:

Res Directory - Select the resource source set where you want to add the image asset: src/main/res, src/debug/res, src/release/res, or a user-defined source set. The main source set applies to all build variants, including debug and release. The debug and release source sets override the main source set

and apply to one version of a build. The debug source set is for debugging only. To define a new source set, select File > Project Structure > app > Build Types. For example, you could define a beta source set and create a version of an icon that includes the text "BETA" in the bottom right corner. For more information, see Configure Build Variants.

The Output Directories area displays the images and the folders where they will appear in Project Files view of the Project window.

Click Finish.

Image Asset Studio adds the images to the mipmap folders for the different densities.

Create an action bar or tab icon

After you open Image Asset Studio, you can add an action bar or tab icon by following these steps:

In the Icon Type field, select Action Bar and Tab Icons.

Select an Asset Type, and then specify the asset in the field underneath:

In the Clip Art field, click the button.

In the Select Icon dialog, select a material icon and then click OK.

In the Path field, specify the path and file name of the image. Click ... to use a dialog.

In the Text field, type a text string and select a font.

The icon appears in the Source Asset area on the right side, and in the preview area at the bottom of the wizard.

Optionally change the name and display options:

Name - If you don’t want to use the default name, type a new name. If that resource name already exists in the project, as indicated by an error at the bottom of the wizard, it's overwritten. The name can contain lowercase characters, underscores, and digits only.

Trim - To adjust the margin between the icon graphic and border in the source asset, select Yes. This operation removes transparent space, while preserving the aspect ratio. To leave the source asset unchanged, select No.

Padding - If you want to adjust the source asset padding on all four sides, move the slider. Select a value between -10% and 50%. If you also select Trim, the trimming happens first.

Theme - Select HOLO_LIGHT or HOLO_DARK. Or, to specify a color in the Select Color dialog, select CUSTOM and then click the Custom color field.

Image Asset Studio creates the icon within a transparent square so there's some padding on the edges. The padding provides adequate space for the standard drop-shadow icon effect.

Click Next.

Optionally change the resource directory:

Res Directory - Select the resource source set where you want to add the image asset: src/main/res, src/debug/res, src/release/res, or a user-defined source set. The main source set applies to all build variants, including debug and release. The debug and release source sets override the main source set and apply to one version of a build. The debug source set is for debugging only. To define a new source set, select File > Project Structure > app > Build Types. For example, you could define a beta source set and create a version of an icon that includes the text "BETA" in the bottom right corner. For more information, see Configure Build Variants.

The Output Directories area displays the images and the folders where they will appear in Project Files view of the Project window.

Click Finish.

Image Asset Studio adds the images in the drawable folders for the different densities.

Create a notification icon

After you open Image Asset Studio, you can add a notification icon by following these steps:

In the Icon Type field, select Notification Icons.

Select an Asset Type, and then specify the asset in the field underneath:

In the Clip Art field, click the button.

In the Select Icon dialog, select a material icon and then click OK.

In the Path field, specify the path and file name of the image. Click ... to use a dialog.

In the Text field, type a text string and select a font.

The icon appears in the Source Asset area on the right side, and in the preview area at the bottom of the wizard.

Optionally change the name and display options:

Name - If you don’t want to use the default name, type a new name. If that resource name already exists in the project, as indicated by an error at the bottom of the wizard, it's overwritten. The name can contain lowercase characters, underscores, and digits only.

Trim - To adjust the margin between the icon graphic and border in the source asset, select Yes. This operation removes transparent space, while preserving the aspect ratio. To leave the source asset unchanged, select No.

Padding - If you want to adjust the source asset padding on all four sides, move the slider. Select a value between -10% and 50%. If you also select Trim, the trimming happens first.

Image Asset Studio creates the icon within a transparent square so there's some padding on the edges. The padding provides adequate space for the standard drop-shadow icon effect.

Click Next.

Optionally change the resource directory:

Res Directory - Select the resource source set where you want to add the image asset: src/main/res, src/debug/res, src/release/res, or a user-defined source set. The main source set applies to all build variants, including debug and release. The debug and release source sets override the main source set and apply to one version of a build. The debug source set is for debugging only. To define a new source set, select File > Project Structure > app > Build Types. For example, you could define a beta source set and create a version of an icon that includes the text "BETA" in the bottom right corner. For more information, see Configure Build Variants.

The Output Directories area displays the images and the folders where they will appear in Project Files view of the Project window.

Click Finish.

Image Asset Studio adds the images in the drawable folders for the different densities and versions

5.Executive Summary

Android users expect your app to look and behave in a way that's consistent with the platform. Not only should you follow material design guidelines for visual and navigation patterns, but you should also follow quality guidelines for compatibility, performance, security, and more

6.Android Launcher Concept

Android UI Controls

Android provides a number of standard UI controls that enable a rich user experience. Designers and developers should thoroughly understand all of these controls for the following reasons:

They are faster to implement. It can take up to ten times longer to develop a custom control than to implement a user interface with standard Android controls.

They ensure good performance. Custom controls rarely function as expected in their first implementation. By implementing standard controls, you can eliminate the need to test, revise and improve custom controls. Moreover, while designers will spend a great deal of time thinking about how a control should look, they may not always consider the many ways in which a custom control will behave in the user's hands. Items on a mobile device often need to grow and shrink in size as they are pinched, or scroll if they are part of a list. As a result, creating a "clean" custom control from scratch can take a significant amount of design and development time. Google, however, has already thought about these interactions and developed standard controls to properly address them.

Android users expect standard controls. Through their interactions with other Android apps, users become accustomed to Android's standard controls. Deviating from the standard Android user experience can confuse and frustrate users, making them less likely to want to use your app and incorporate it into their daily activities.

With a solid awareness of Android's standard controls, designers and developers can speed app development while offering users an intuitive experience that feels instantly familiar.

Activities

Android applications are composed of "activities" which are unique, focused actions a user can take. Because it can be difficult or time-consuming to scroll, zoom in, or click links on a small screen, it is recommended that an app display only one activity per screen. This practice presents the user with only the most relevant information and allows them to launch a new screen for additional

information, or click the “back” button to view the previous activity. While a screen can expose multiple tasks, it should help the user complete just one activity at a time.

In Gmail for example, a user can only read the body of an e-mail (right) once he has clicked the relevant message (left). This layout reduces the amount of information displayed on each screen and allows the user to easily navigate between the Inbox and the message text.

User Interactions

When a user first downloads your application, he will make snap judgments on the usability and intuitiveness of the application within the first few minutes of use. It is, therefore, crucial to balance the creativity of your app with the standard user interactions Android users have come to expect. These include:

Hard buttons: including Back, Menu, Home and Search buttons. Soft buttons that duplicate these features will only confuse or frustrate Android users. Moreover, back button behavior can be tricky and needs to be defined up-front for every screen, as it is not always as simple as returning to the previous activities. Most mobile phones, for example, offer both an “incoming call” activity and an “active call” activity. Once a user has answered and completed the call, the user would not expect to return to the “incoming call” activity upon pressing the “back” button, but rather to the activity that occurred before the incoming call. If the app offers only one activity, the back button should return the user to the device’s home page.

Long press elements: Items of a list can be long pressed to open a context menu that provides secondary information. “ToDo” list apps, for example, often use a touch interaction to mark a task as completed and a long press interaction to display a menu with “edit” or “delete” functionality.

Layouts

Android UI screens are frequently resized, both on the fly via pinch and zoom as well as at startup when Android adjusts the size of the UI to fit the screen size of the mobile device on which it’s running. In order to make the most of the screen size and handle this resizing gracefully, Android provides a number of screen layout options.

First, Android developers must specify whether each screen should follow a linear layout which manages controls in a horizontal or vertical fashion or a relative layout which manages controls in relation to one another. Linear layouts are the most common, as in the example below. At left, the controls only stretch to accommodate the text and are positioned in a horizontal line. In the middle image, the same rules apply but in a vertical layout. At right, the vertical layout is maintained but the middle button stretches to accommodate the screen rather than the text.

A relative layout defines the position of controls by their relationship to other components on the same screen. In the example below from the droidcake.com blog, the “OK” button was specified to be set below the radio button group. The “Cancel” button was specified to be set to the right of the OK button with its right edge extended to the edge of the screen. This relative layout positioning ensures the position of the buttons remains constant across a variety of screen sizes.

Android also offers specific layout properties to control the way in which screen elements are displayed across Android devices and during use:

Weight: The weight property allows the developer to determine how free space is divided on the screen.

Gravity: Gravity is the term used for control alignment (right, bottom, top, or left) on an Android device.

Density independence: Your application achieves “density independence” when it preserves the physical size (from the user’s point of view) of user interface elements displayed on screens with different densities. Without density independence, a UI element (such as a button) will appear larger on a low-density screen and smaller on a high-density screen.

So who specifies all of these properties?

If an Android application is designed in a vacuum and then “thrown over the wall” to the development team, you must rely on the developers’ interpretation of the design which may vary significantly from the original intent. On the other hand, the development team shouldn’t be expecting the designer to specify the weight, gravity and other layout properties of each screen and control.

In our experience, the best practice is to have the designer document the layout and resize behavior of each screen to the development team via a series of wireframes, if not a full style guide. The designer should then stay in close communication with the development team as the developers work to determine the right combination of Android layout properties to realize the design.

Screen Size

A common misconception is that an Android app should be designed to support only a specific set of Android devices. Many teams assume their app will only look right on a screen of a particular screen size and limit their design to suit only a handful of devices supporting that size. In reality, Android

offers you tools needed to develop a visually impressive interface that supports the full range of devices and screen sizes on the market.

To help you accommodate the range of Android screen sizes, Android recommends designing four versions of the application UI:

A small version for screens under 3".

A normal version to accommodate 3" to 4.5" screens.

A large version for viewing on 4.5" to 10" screens.

An extra large version for devices with screens larger than 10" (tablet).

It is not strictly necessary to create a design for all four versions – in some cases; one "normal" and one "extra large" version may suffice. If, however, you need to display a large number of controls on your screen, or your organization wishes to ensure perfect consistency across screen sizes, you may decide to accommodate all four size categories listed above.

Fragments

A smartphone should only display one activity per screen due to its small screen size. Tablet devices, however, offer additional screen real estate and are often used in a similar setting as a desktop or notebook, meaning the application could show more information at once on the screen. Using an Android construct called fragments, designers and developers can merge portions of the UI onto one large screen or split them into individual screens for use on small screens. This can help to reduce the number of interactions a user must perform on a device with a large screen and eliminate wasted space.

The example below shows a Gmail interface on a tablet display. This design uses fragments to display both the navigation list at left and the Inbox content at right. The design reduces the number of screens that must load before the user reaches the desired message.

If you anticipate your app will someday be used on a tablet device, we strongly recommend you incorporate fragments into your design. Designers need to be aware of the concept of fragments in order to design by fragment, and developers also need to be aware of this concept and its implementation details.

By designing custom, reusable fragments for each screen activity at the beginning of the project, you can eliminate the need to create an entirely new layout for a tablet device.

Intents

Android applications typically borrow from other applications already on the device. Using intents you can simplify both the programming requirements for your app and offer simpler, less cluttered screens.

If your app needs to perform a function beyond its core abilities such as opening a photo, looking up a contact, or playing a video, the team should investigate whether a tool that can perform that function already exists in the OS or in a popular third-party app. If so, you can leverage that functionality using intents.

For example, if your app accesses user contacts, you can use intent objects to launch the device's existing Contacts application. This will eliminate programming duplication and speed up the user's interaction with the device since the user will not need to re-learn how to add a contact to your particular app.

Android offers specific UI controls, activities, interactions, layout and resize options, as well as special constructs like fragments and intents. While on the surface these appear to be things that the design team needs to work with, we contend that the entire team must be immersed in Android to coordinate design, workflow, and execution into a single, intuitive application — one that grabs users' attention and draws them into the real value of your product.

7. Recommendations

Recommendations help users quickly find the content and apps they enjoy. Creating recommendations that are high-quality and relevant to users is an important factor in creating a great user experience with your TV app. For this reason, you should carefully consider what recommendations you present to the user and manage them closely.

Types of recommendations

When you create recommendations, you should link users back to incomplete viewing activities or suggest activities that extend that to related content. Here are some specific type of recommendations you should consider:

Continuation content recommendations for the next episode for users to resume watching a series. Or, use continuation recommendations for paused movies, TV shows, or podcasts so users can get back to watching paused content in just a few clicks.

New content recommendations, such as for a new first-run episode, if the user finished watching another series. Also, if your app lets users subscribe to, follow, or track content, use new content recommendations for unwatched items in their list of tracked content.

Related content recommendations based on the users' historic viewing behavior.

For more information on how to design recommendation cards for the best user experience, see Recommendation Row in the Android TV Design Spec.

Refresh recommendations

When refreshing recommendations, don't just remove and repost them, because doing so causes the recommendations to appear at the end of the recommendations row. Once a content item, such as a movie, has been played, remove it from the recommendations.

Customize recommendations

You can customize recommendation cards to convey branding information, by setting user interface elements such as the card's foreground and background image, color, app icon, title, and subtitle. To learn more, see Recommendation Row in the Android TV Design Spec.

Group recommendations

You can optionally group recommendations based on recommendation source. For example, your app might provide two groups of recommendations: recommendations for content the user is subscribed to, and recommendations for new trending content the user might not be aware of.

8. Conclusion

I've learned through my research that Android is a much more diverse operating system than iOS and Windows Phone Mobile. Android has grown rapidly over the past 4 years becoming the most used smartphone operating system in the world. It's because Android doesn't release 1 phone from 1 company with 1 new OS every year, but countless phones from numerous companies, adding their own twist, throughout the year, developing gradually day-by-day. Android's ability to customize is unparalleled compared to Apple's and Microsoft's software allowing the user to change and customize nearly every aspect of Android which most iPhone and Windows 7 users wouldn't dream possible. I am not one to say that Android is better or worse than one OS, but is unique and incomparable to other mobile operating system