GitHub Username: szekelyistvan23

Color Palette

Description

The app displays two types of list with top and new palettes to choose from and you can save them in a favorite list. When the user clicks on one of the palettes, he will receive more detailed information about it.

Intended User

The app helps mobile app developers and designers to choose a palette or a good color combination. It is a source of inspiration to make web-pages and mobile apps more colorful, beautiful and more appealing to the user.

Features

- Displays color palettes
- Gives details about palettes, such as color's hex code, original web-page's link
- Shares palettes' detail data through SMS, email, etc.
- Saves them to a favorite list
- Displays a widget with color palettes

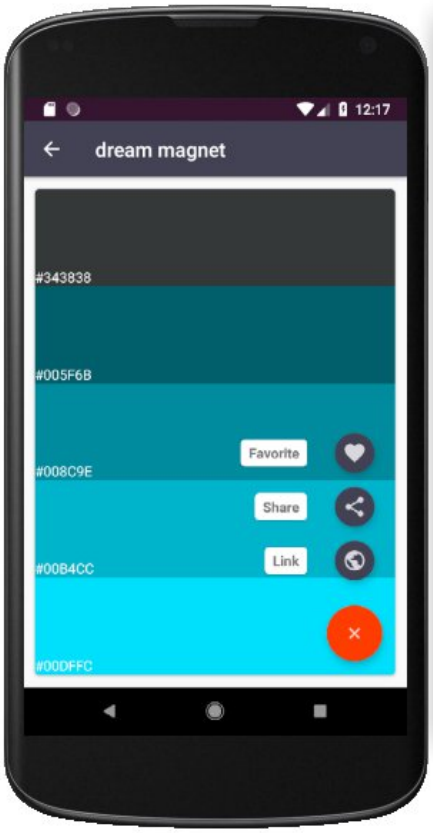
User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1

The main screen of the app with bottom navigation to choose from three types of list to be displayed. A palette contains five or four colors and on the right edge is a label with the name of the palette.

Screen 2



The second screen of the app, with more details about the palette. Displays the colors and their hexadecimal codes. Another feature of this screen is the FAB menu with three options. The user can add the palette to the favorite list, share the color codes and view the page of the palette on the web with additional pieces of information.

Widget

The widget of the app contains a list of palettes, the user can scroll them and if he taps on the palette an activity will be opened with detailed information about the chosen palette.

Key Considerations

Meets project requirements?

- App is written in Java
- Enables RTL layout switching, all the layouts support it
- All the string constants are stored in strings.xml, custom colors in colors.xml, custom themes in styles.xml and layout dimensions in dimens.xml
- Android Studio version 3.1.3
- Gradle version 3.1.3

How will your app handle data persistence?

The app uses a Content Provider with three databases for each of the app's list: top, new and favorite. At the first start, the app connects to the Internet and downloads the palettes data from COLOURlovers.com to the top and new databases.

Describe any edge or corner cases in the UX.

On the detail page, if the FAB menu is open and the user clicks the back button, the up button or makes a swipe the FAB menu will close.

Describe any libraries you'll be using and share your reasoning for including them.

Retrofit is a type-safe HTTP client for Android and Java – developed by Square, I use it to download JSON data from the Internet and automatically deserializing it.

ButterKnife is a view "injection" library for Android which uses annotation processing to generate boilerplate code for you. I use it to eliminate the findviewbyid() method from my code.

Floating Action Button Speed Dial it is an Android library providing an implementation of the Material Design Floating Action Button Speed Dial. This is a feature that I use in the detail fragment.

Libraries used and their versions:

- Support Library v7:27.1.1

- Retrofit2: 2.4.0
- ButterKnife: 8.8.1
- SpeedDial: 1.0.1
- Firebase Core: 16.0.1
- Google Play Services Ads: 15.0.0

Describe how you will implement Google Play Services or other external services.

Google Mobile Ads SDK, I use it to display an interstitial ad between the two activities. When the user clicks on a palette an ad will show up and after that, he will see the detail activity with the palette details.

Firebase Crashlytics, is a lightweight, real-time crash reporter that helps me to track, prioritize, and fix stability issues that erode my app's quality.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

- Adding dependencies
- Adding custom drawables

Task 2: Implement MainActivity UI

- Build MainActivity's layout
- Implement a RecyclerView
- Implement a BottomNavigationView
- Implement a OnOptions menu to exit from the app

Task 3: Implement DetailFragment UI

- The fragment's activity implements a ViewPager
- Build DetailFragment's layout
- DetailFragment contains a FAB menu

Task 4: Add a Content Provider

- Creating classes needed for a Content Provider
- Creating a separate table for each list of the app
- Implement AsyncTaskLoaders to handle Content Provider operations

Task 5: Add a Service

- Setting up an Intent Service to download data from Internet
- The service will start at first run and stops when data download is successful

Task 6: Add a widget

- Setting up a widget
- Link widget to Content Provider
- The widget contains the list of top palettes

Task 7: Add an interstitial ad

- Setting up an ad
- Displays the ad when going from MainActivity to DetailActivity

Task 8: Add a FAB menu

- Setting up a FAB menu
- The menu contains the following sub-menus: Favorite, Share, Link
- When Favorite is selected: the color palette will be added or removed from the favorite table
- When Share is selected: the color palette's data will be shared through an SMS, email, etc.
- When Link is selected: the palette's web-page will be opened