Developing First Android App

REVIEW

CODE REVIEW 6

HISTORY

Meets Specifications

Great work really

well-formatted code along with a smoothly working app with nice UI and transition animations, you did a great job here really.
MVVM implemented correctly along with efficiently using data-binding, also navigation works great.

I really enjoyed reviewing your project, thank you.

Congratulations 🏆

Code Quality

✓

The Detail screen needs to add the new item to the view model. The listing screen should be listening to that model and show the new item.

- Well-done implementing activity-scoped shared ViewModel between fragments,these fragments can share a ViewModel using their activity scope to handle this communication,Shoe list fragment updates correctly upon adding new shoe item in details fragment. 👌
  further resources:
  - Shared ViewModel in Android: Shared between Fragments
  - Share data between fragments

✓

There should only be one activity: MainActivity. Each screen should be a fragment.

- Really good, Single-activity multiple fragments approach implemented very well, few resources to discuss this approach vs multiple activities:
  - A Single-Activity Android Application. Why not?!
  - Reasons to use Android Single-Activity Architecture with Navigation Component
  - 10 best practices for moving to a single activity

✓

The project's code is error-free.

app builds and runs without errors, 👏

Layouts

✓

The project correctly implements LinearLayout and ConstraintLayout to match the complexity of the layout of a page. Using code comments, the project justifies the use of ConstraintLayout or LinearLayout

- Great,ConstraintLayout allows you to create large and complex layouts with a flat view hierarchy (no nested view groups). It's similar to RelativeLayout in that all views are laid out according to relationships between sibling views and the parent layout, but it's more flexible than RelativeLayout and easier to use with Android Studio's Layout Editor.
  here are a couple of resources to master ConstrainLayout:
  - ConstraintLayout Tutorial for Android: Complex Layouts
  - 5 tips to master ConstraintLayout

✓

1. **All layouts will use the <layout> tag to support Databinding.**
2. **Detail screen uses the <data> element.**
3. **Databinding is set to the appropriate setting in the app build.gradle file**

- All layouts use the `<layout>` tag..
- Details fragment layout uses the ShoeDetailViewModel `<data>` element.
- Databinding is enabled correctly in app `build.gradle` file
- DataBinding used effeciently, it helps binding UI components in your layouts to data sources in your app using a declarative format rather than programmatically.
  - Advanced Android Data Binding by Jose Alcérreca, Google EN

✓

The detail layout contains an <data> element with the name of the variable and the class associated with it.

- All EditViews correctly refer to created class variable

two-way databinding implemented very-well, here are a couple of resources besides the official guides

- Android MVVM LiveData Data Binding
- Two-way Android Data Binding
- Advanced Data Binding: Binding to LiveData (One- and Two-Way Binding)

- The app contains at least 5 screens.
- The app contains correctly laid-out labels and edit fields for each screen.
- The app contains button positioned below the text fields

1. A new item layout is created for each item
2. New item layout is added to LinearLayout
3. Layout is updated with items added on the detail screen

Really good utilizing also databinding for shoe item layout inflation, few students prefer this approach, most use the old way of creating `new View()` s. well-done 👏🏽

1. Layout created with a label & edit view for each item
2. Uses data binding to save data
3. Uses a save button to save data to view model

Navigation

The app needs to traverse the following screens in the correct order:

- Login
- Welcome
- Instructions screen
- Listing screen
- Detail screens
  The app should also be able to navigate back via the back arrow or the back button.
- A navigation file has been created that defines a start destination.
- All destinations have a fragment, label and action associated with it.

App flow works as expected with great transition animations upon the screen transitions, also you can utilize shared elements transitions.
a couple of resources for transitions and shared element transitions with navigation component :

- Custom Transition Animations with Navigation Component | Android Studio Tutorial
- Transitions in Android Navigation Component

✓

1. All code will use the DataBindingUtil class to inflate the layout.
2. All click listeners are connected via the DataBindingUtil class and uses the NavController to navigate to the next screen.

✓

This menu will appear only on the Shoe Listing screen and will return the user to the login screen

DOWNLOAD PROJECT

6 CODE REVIEW COMMENTS ›

RETURN TO PATH

Rate this review

START