Task Instructions

1. Install Android Studio
    a. Before you begin writing code for this task, you're going to need an up-to-date version of Android Studio. Android Studio is the official integrated development environment (IDE) for Android development. It is packed with features to help you develop apps, from project scaffolding to code completion. It also comes with a built-in emulator, so don't worry if you don't have a physical Android device to work with, you can test your app using your computer. For installation instructions, follow this guide.
        i. Note: At the time of writing, there is a particularly pesky bug in Android Studio Dolphin that prevents some Backpack components from being displayed in the design section of the editor. If you encounter this issue, please try a different version of Android Studio (such as Electric Eel). The bug tracker for this issue can be found here.
2. Create a new project
    a. Now that you have Android Studio installed, it's time to create a new project! Open Android Studio and click the 'New Project' button. Next, it's time to choose a starting point. For this task, we'll be using the 'Empty Activity' template in the 'Phone and Tablet' section. You'll need to tell Android Studio a little more about your project — come up with a catchy name and pick a reasonable location to save it. For the language, choose 'Kotlin' and for minimum SDK choose 'API 33: Android Tiramisu'. Don't worry about the package name, it should be auto-generated from the project name. When you're ready, click 'finish' to initialise your app!
    b. If this is your first time working with Android Studio, take some time to poke around the IDE and explore what it has to offer. There's far too much to cover over the course of a single tutorial, and the sheer number of options available can be a little overwhelming, but don't be discouraged, we'll walk you through everything you need to know to get an application up and running.
3. Run your application
    a. As soon as you create your project, Android Studio will begin initialisation. Wait until the progress bar in the bottom right corner completes, after which you can launch your app on the built-in emulator. Click the 'run' button (green arrow) in the upper right corner next to the devices dropdown to start. When you do so, Android Studio will build your application, launch the emulator and execute your code. You should see a virtual phone appear on the right side of your screen. If your emulator isn't working, consult this list of troubleshooting steps to help you along your way.
4. Add Backpack as a dependency
    a. Now that your app is working, it's time to integrate the Backpack UI library. Android projects use the Gradle build system in order to integrate dependencies, automate all sorts of monotonous tasks and actually build your projects. In order to add Backpack, you'll need to find and open your module's build.gradle file

(which can be found under the ‘Gradle Scripts’ section of the project sidebar on the left side of Android Studio). Add the following line to the ‘dependencies’ block of this file: ‘implementation 'net.skyscanner.backpack:backpack-android:43.0.0'’. A dialog box should appear at the top of the screen letting you know that Gradle needs to sync. Click ‘Sync now’, or find the sync button in the Gradle sidebar on the right side of Android Studio. This will refresh Gradle and bring it up to date with the configuration file you just edited.

5. Build the UI
    a. Now it’s finally time to add some Backpack components to your new UI! What our app looks like is determined by an xml file called ‘activity_main’. This file describes all of the different components and their corresponding parameters on the screen.
    b. Open up ‘activity_main.xml’ from the ‘res/layout’ folder of the project sidebar. There are three ways to view this file, listed in the upper right corner. ‘Code’ will show you the raw xml code; ‘Design’ will let you view and edit visual components on the screen; and ‘Split’ will give you a hybrid view.
    c. Let’s start by deleting the greeting component that was automatically generated at project creation, ‘hello world’. You can do so by deleting the xml snippet or by deleting the visual component in the Design pane.
    d. With our newly blank slate, it’s time to add some of our own components. Let’s start with a card — take a look at the Backpack documentation. Copy the xml snippet into your code as a child of the ‘ConstraintLayout’ tag. This component is now floating casually around the screen and needs to be constrained to a location. From the ‘Component Tree’ sidebar of the Design pane, right click on ‘BpkCardView’ and centre the component horizontally and vertically. The red squiggles under your card component in the Code view should now be gone, since the component has a defined location on the screen. Next, let’s embellish your card by enlarging the corners — change the ‘cornerStyle’ attribute to large from either the Code view or the attributes sidebar of the Design pane.
    e. Now let’s add some information to the UI. Change the ‘android:text’ attribute of the ‘BpkText’ component to read ‘Departure’. Then add the following attribute to the component in order to increase its size: ‘style="@style/bpkTextHeading1"’.
    f. Now for some more structure! Add a ‘LinearLayout’ (vertical) to your UI by dragging it from the Palette sidebar of the Design pane into the Component Tree underneath ‘BpkCardView’. Then move the ‘BpkText’ component into the layout.
    g. You should now have a pretty good sense of how to manipulate components and their attributes using Android Studio. Your task is to create a UI that incorporates the following information in a visually pleasing fashion:
        i. Flight information card with:
            1. Flight number
        ii. Departure card with:
            1. A three-digit airport code of your choice
            2. A departure time
        iii. Arrival card with:

        1. A three-digit airport code of your choice
        2. An arrival time

    h. Once you are happy with the way your UI looks, hit the run button again to see it in action! If everything looks good, you’ve completed the task and all that remains is to submit your work.

6. Submit your work
    a. Push your code to a public github repository and submit a link to your repo below. If you are unfamiliar with git, read through the first two chapters of the git book to learn the basics.