Practice: Build Sports app

1. Before you begin

Congratulations! In this pathway, you learned how to make your app adaptive by using the `WindowWidthSize` class and the canonical layout. Now it is time to put what you learned into practice.

In this practice set, you will build on the concepts you learned in this pathway by creating a Sports app. The starter app is fully functional for a mobile layout. Your task is to make it adaptive for large screens.

The solution code is available at the end, but try to solve the exercises before you check it. Work through the problems at a pace that's comfortable for you. There are durations listed, but you do not have to adhere to them because they're only estimates. You're encouraged to take as much time as you need to solve each problem thoughtfully.

Prerequisites

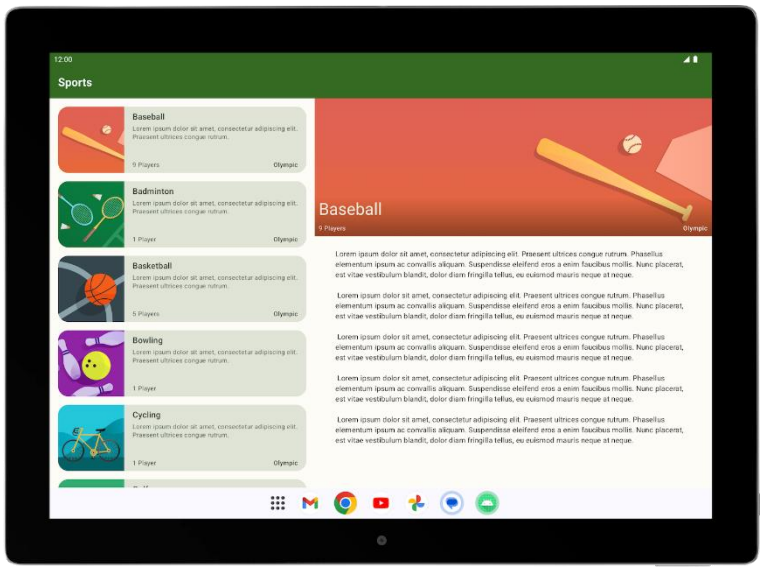
- Complete Android Basics in Compose coursework through the Build an adaptive app with dynamic navigation and Build an adaptive app with adaptive layout codelabs.

What you'll need

- A computer with internet access and Android Studio installed
- Access to GitHub

What you'll build

A Sports app that adapts to a large screen. The finished app looks like the following image:



2. Get started

Download the starter code

In Android Studio, open the `basic-android-kotlin-compose-training-sports` folder.

Starter code URL:

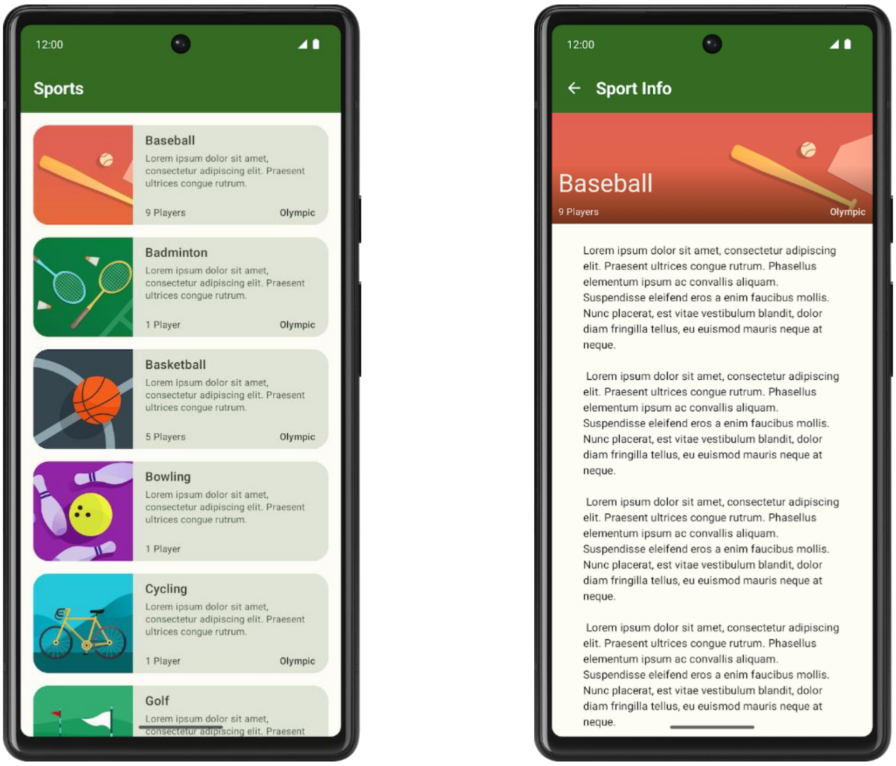
https://github.com/google-developer-training/basic-android-kotlin-compose-training-sports/tree/starter

Branch name with starter code: `starter`

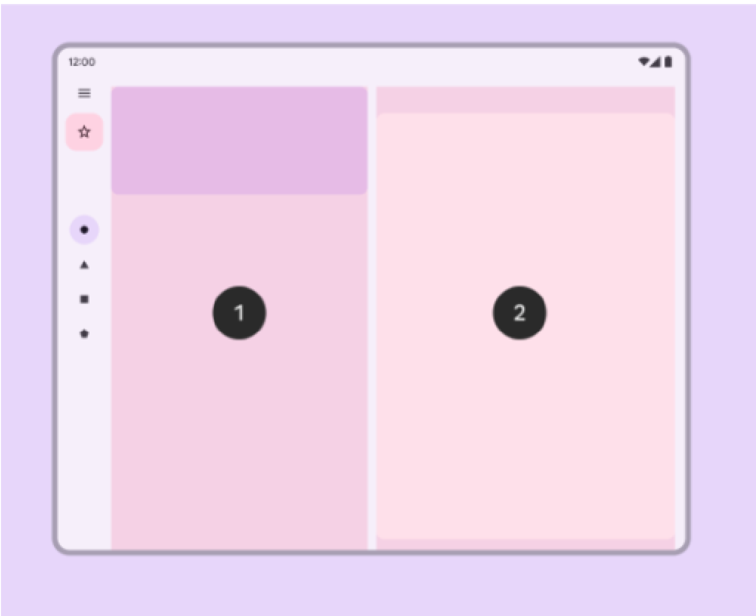
3. Plan to adapt Sports app for large screens

To adapt the Sports app for large screens, you first want to establish the type of layout that best displays the app on large screens.

1. Start reviewing the current layouts available on the compact screen size. There are two screens — namely the list screen, which corresponds to the `SportsList` composable, and the detail screen, which corresponds to the `SportsDetail` composable.



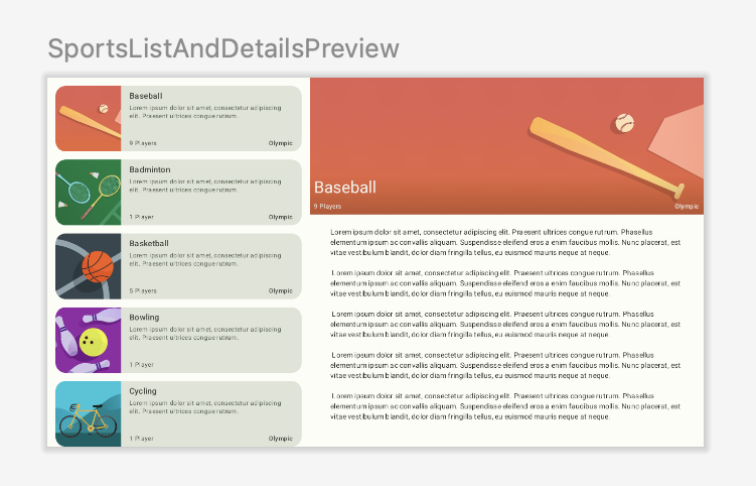
2. Review the canonical layout (https://m3.material.io/foundations/adaptive-design/canonical-layouts) to determine the layout that best fits the Sports app use case.
3. Sketch out a possible expanded screen layout. Use either a simple visual design software or a piece of paper to visualize how the screens fit together in the expanded layout.



4. Create the expanded screen in code

Now that you have a clear view of how the expanded layout looks, let's translate that into code.

1. Create a composable for the expanded screen that shows both the list and the details screen. You can name it `SportsListAndDetails` and place it in the `SportsScreens.kt` file.
2. Create a composable preview to simplify verification of the UI for the `SportsListAndDetails` composable.



3. Ensure that the back button behavior is appropriate for the expanded screen. When the user presses the back button, you want them to exit the app when the main screen appears. You can change this behavior by passing the appropriate lambda to the `SportsDetails` composable. **Hint**: you can have access to the app's `Activity` from `(LocalContext.current as Activity)`.

5. Make app change to different layout based on window size

To add the expanded screen composable to the app, complete the following tasks:

1. Add `androidx.compose.material3:material3-window-size-class` to the app's `build.gradle.kts`.
2. Calculate `windowSizeClass` in the `MainActivity` and pass the `widthSizeClass` to the `SportsApp` composable.
3. Create a new directory named `utils` and create a new file name `WindowStateUtils.kt`.
4. Add an `enum` class in `WindowStateUtils` to denote two different `contentType`. You can name them `ListOnly` and `ListAndDetail` types.

5. In the `SportsApp` composable, determine the `contentType`, based on the `widthSizeClass`.
6. Display the `SportsListAndDetails` composable when `contentType` is `ListAndDetail` and keep the previous behavior when the `contentType` is `ListOnly`.
7. For the `SportsAppBar` composable, change the behavior so that the back button doesn't appear and the app bar shows **Sports** when the screen expands in the list page.
8. Verify the UI and navigation capabilities for both compact and expanded screens using the resizable emulator.

6. Get the solution code

To download the code for the finished codelab, you can use this git command:

```
$ git clone https://github.com/google-developer-training/basic-android-kotlin-compose-training-sports.git
```

Alternatively, you can download the repository as a zip file, unzip it, and open it in Android Studio.

https://github.com/google-developer-training/basic-android-kotlin-compose-training-sports/archive/refs/heads/main.zip

Note: The solution code is in the `main` branch of the downloaded repository.

If you want to see the solution code, view it on GitHub.

https://github.com/google-developer-training/basic-android-kotlin-compose-training-sports/tree/main