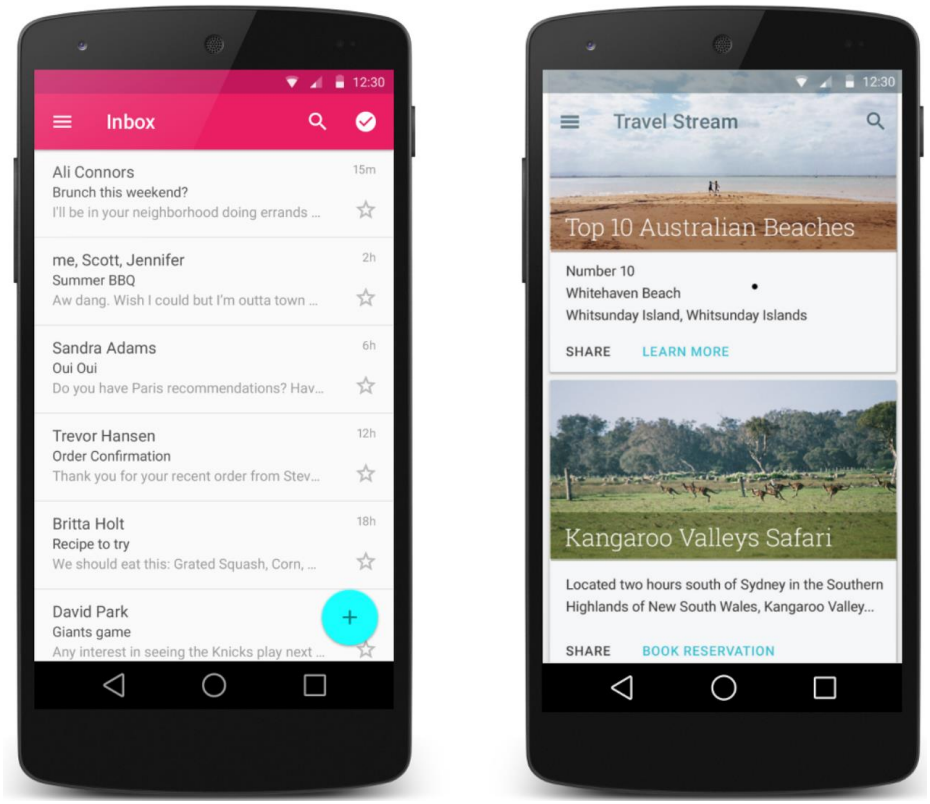


Lists & Grids

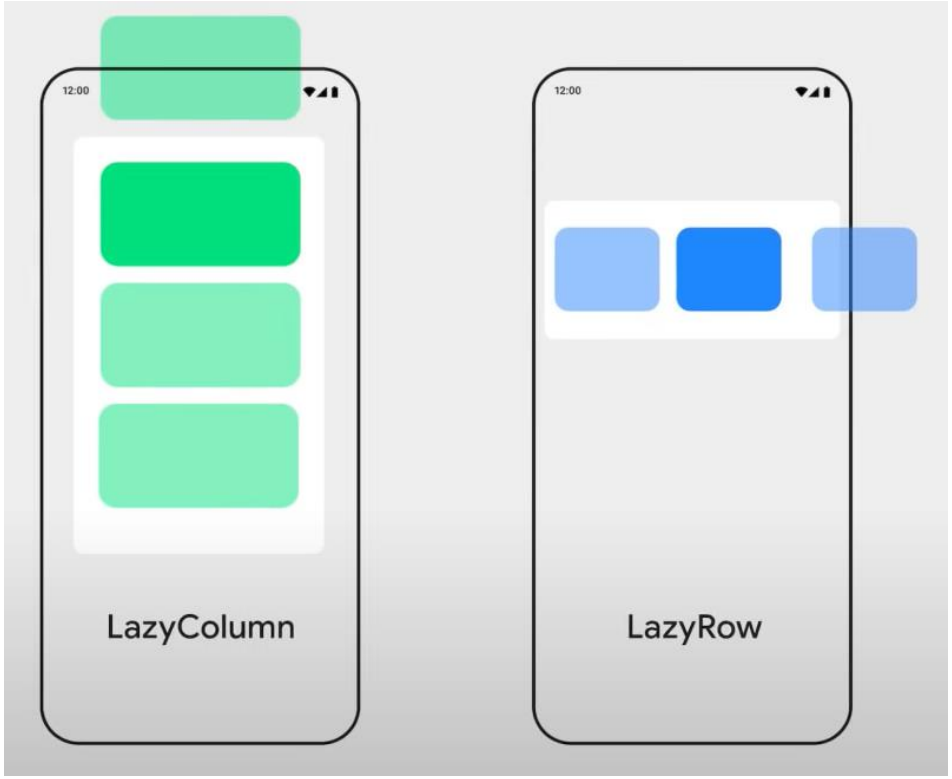
Dr. Abdelkarim Erradi

CSE@QU

Outline

1. Displaying a List
2. Interacting with a List
3. Displaying a Grid

Displaying a List



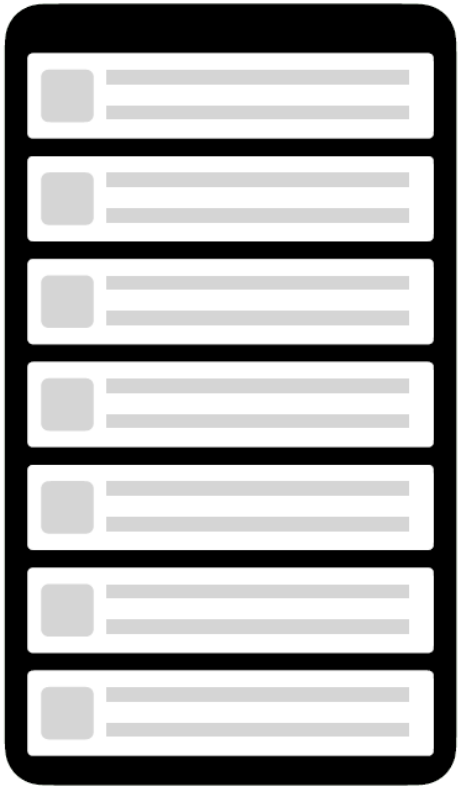
Back

Displaying a List

- In apps it is common to display collections of items
- For displaying a small collection of items, a **`Column`** or **`Row`** layouts could be used
  - The **`verticalScroll()`** modifier could be applied to make the `Column` scrollable
  - The **`horizontalScroll()`** modifier could be applied to make the `Row` scrollable
- For displaying a large list, using a Column/Row layout can cause performance issues
  - Since all the items will be composed and laid out whether or not they are visible
  - Use a Lazy List (i.e., LazyColumn or LazyRow) to only compose and lay out items which are **visible on screen**

Displaying a List

Making the Column scrollable by using the `verticalScroll()` modifier



```
@Composable
fun SurahsList(surahs: List<Surah>) {
    Column(modifier =
     Modifier.verticalScroll(rememberScrollState())
    )
        if (surahs.isEmpty()) {
            Text("Loading surahs failed.")
        } else {
            surahs.forEach {
                SurahCard(surah = it)
            }
        }
    }
}
```

Common Modifiers

- `Column(modifier = Modifier.verticalScroll(rememberScrollState())`

Makes the column scrollable

- `Row(modifier = Modifier.horizontalScroll(rememberScrollState())`

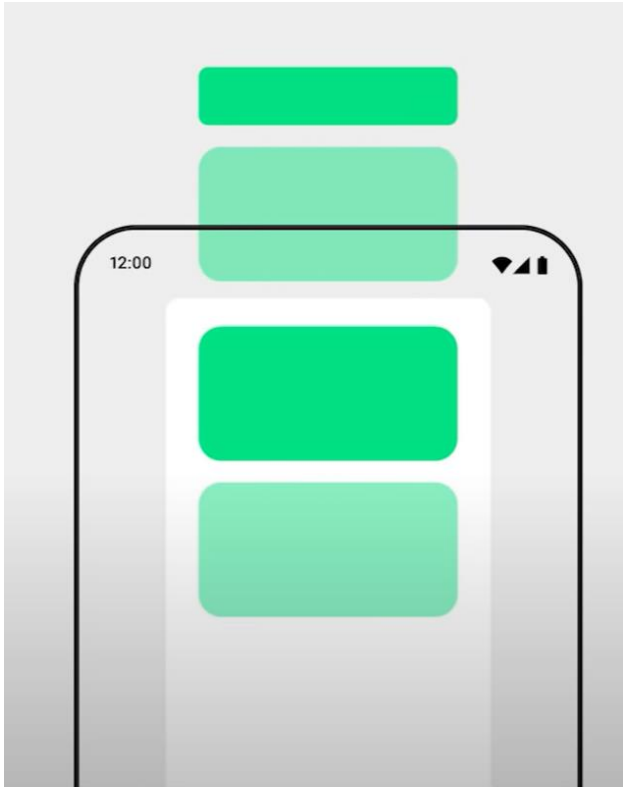
Makes the row scrollable

- `Modifier.fillMaxWidth() / .fillMaxHeight() / .fillMaxSize()`

  occupy the available space

What is a Lazy List?

- A Lazy List is a scrollable container for displaying a list of composables
  - **LazyColumn** produces a vertically scrolling list, and **LazyRow** produces a horizontally scrolling list
- A flexible container for efficiently displaying, and interacting with large sets of data
  - As user scrolls, composables are created to display new items
  - Efficient as it uses a limited number of composables



Lazy List methods

- Lazy List provides several functions for describing items in the layout:
  - **`item()`** to add a single item (e.g., header/footer)
  - **`items(list)`** to add multiple items
  - **`itemsIndexed(list)`** to add multiple items and provides an index

```
import androidx.compose.foundation.lazy.items
…
LazyColumn {
    items(surahs) {
        SurahCard(it)
    }
}
```

Spacing List Items

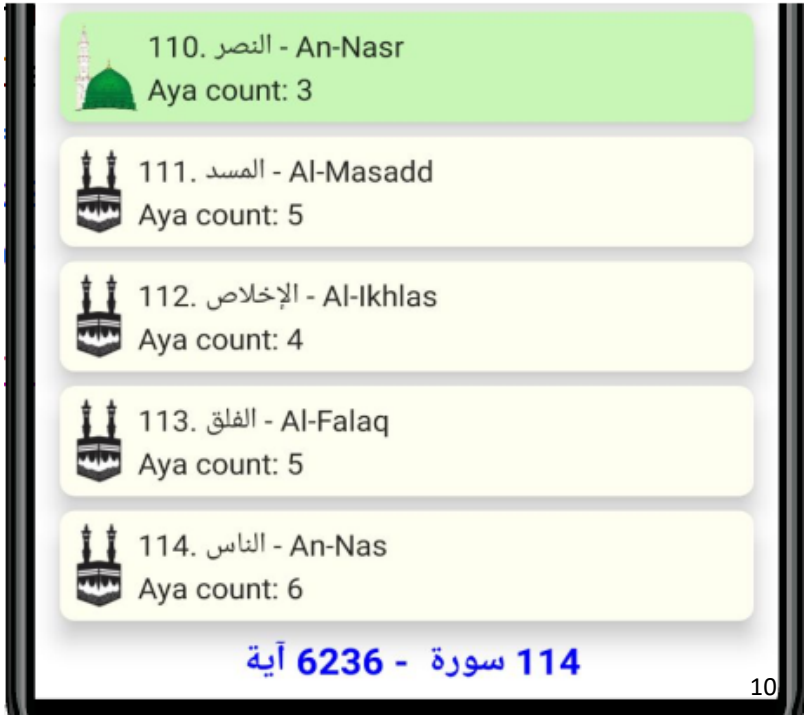
- Use **Arrangement.spacedBy()** to add spacing in-between items

```
LazyColumn(
    verticalArrangement = Arrangement.spacedBy(8.dp),
) {}
```

- Similarly, for LazyRow:

```
LazyRow(
    horizontalArrangement = Arrangement.spacedBy(8.dp),
) {}
```

```
LazyColumn(contentPadding =
    PaddingValues(horizontal = 8.dp, vertical = 8.dp),
    verticalArrangement = Arrangement.spacedBy(8.dp)
) {
    item {
        Text(
            text = "سور القرآن الكريم",
            textAlign = TextAlign.Center,
            modifier = Modifier.fillMaxWidth(),
            style = TextStyle(
                fontWeight = FontWeight.Bold,
                fontSize = 24.sp,
                color = Color.Blue
                textDirection = TextDirection.Rtl
            )
        )
    }
    items(surahs) {
        SurahCard(it)
    }
    item {
        Text(
            text = "$surahCount سورة  - $ayaCount آية",
            textAlign = TextAlign.Center,
            modifier = Modifier.fillMaxWidth(),
            style = TextStyle(
                fontWeight = FontWeight.Bold,
                fontSize = 20.sp,
                color = Color.Blue,
                textDirection = TextDirection.Rtl
            )
        )
    }
}
```

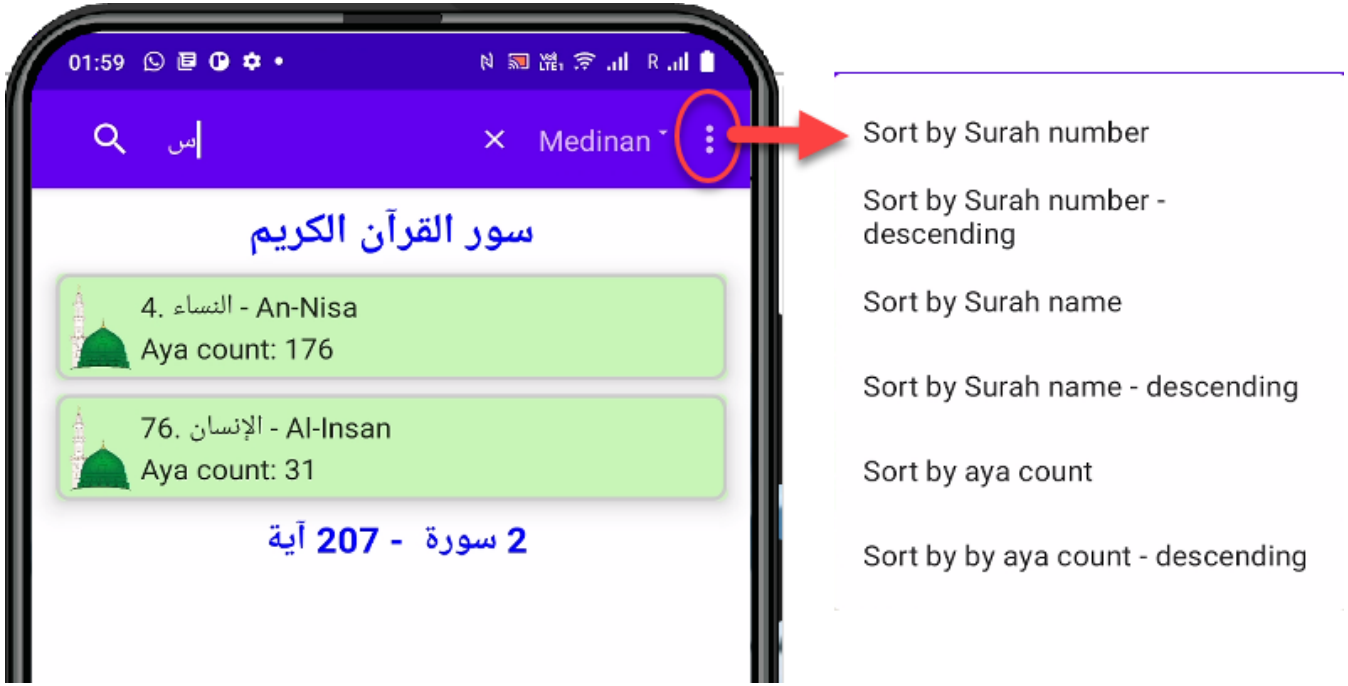


Interacting with a List

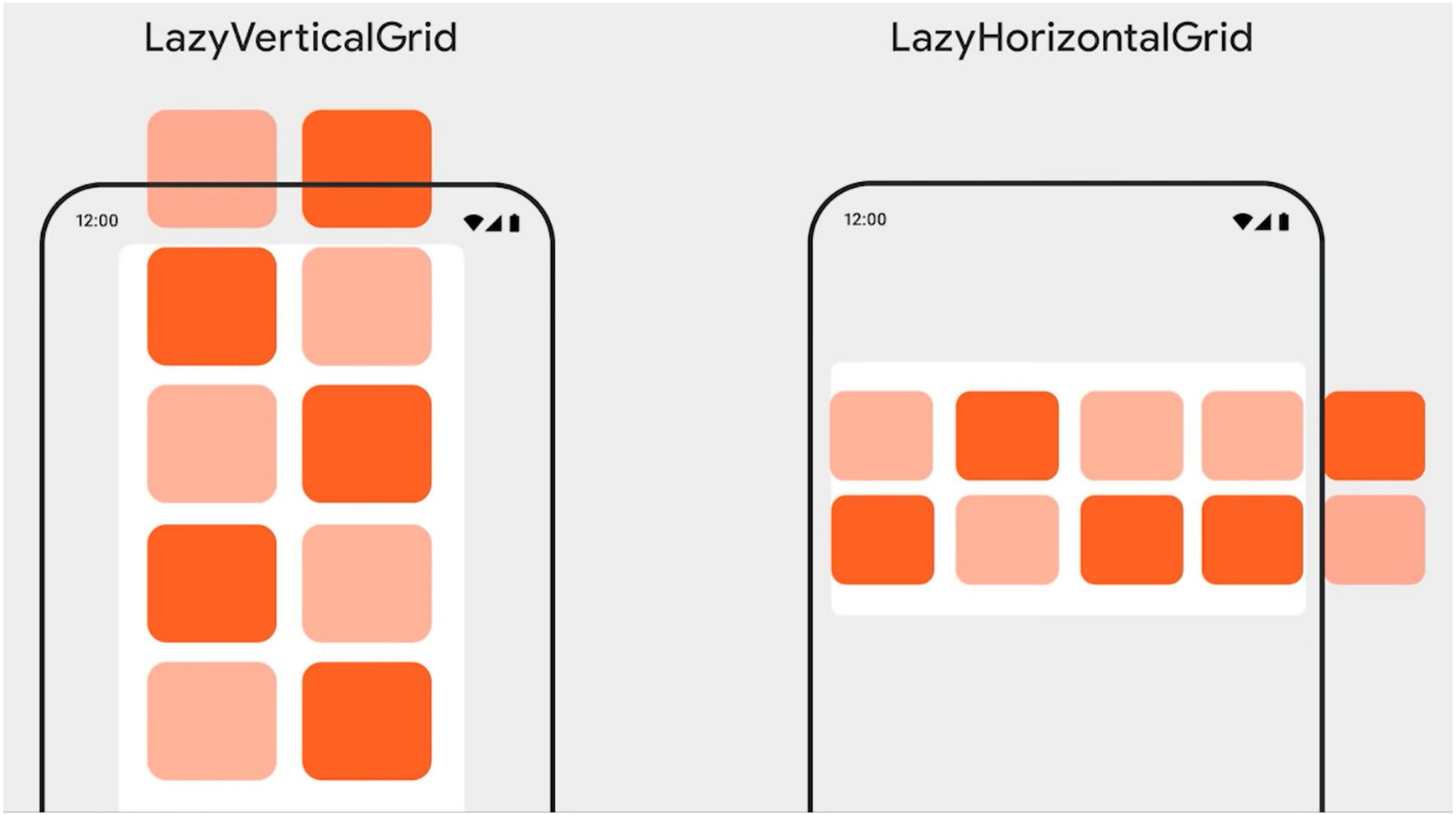
Search & Sort

- Add a SearchBox to the top App Bar
- Add sort options as menu items to the App Bar
- Handle search and sort in the composable displaying the list

See posted Surah example



Displaying a Grid



Back

Lazy grids

- **`LazyVerticalGrid`** and **`LazyHorizontalGrid`** allow displaying items in a grid
  - A Lazy vertical grid displays its items in a vertically scrollable container, spanned across multiple columns
  - A Lazy horizontal grids will have the same behavior on the horizontal axis
- The **`columns`** parameter in LazyVerticalGrid and **`rows`** parameter in LazyHorizontalGrid control how cells are formed into columns or rows
  - **`GridCells.Fixed(count)`** specified the number of columns to be used
  - **`GridCells.Adaptive(minSize = 200.dp)`** lets you specify a width for items, and then the grid will fit as many columns as possible

GridCells.Adaptive

- **`GridCells.Adaptive(minSize = 200.dp)`** lets you specify a width for items, and then the grid will fit as many columns as possible.
  - Any remaining width is distributed equally among the columns, after the number of columns is calculated
  - This adaptive way of sizing is especially useful for displaying sets of items across different screen sizes

Lazy staggered grid

- **`LazyVerticalStaggeredGrid`** and **`LazyHorizontalStaggeredGrid`** are composables that allow you to create a lazy-loaded, staggered grid of items
  - A lazy vertical staggered grid displays its items in a vertically scrollable container that spans across multiple columns and **allows individual items to be different heights**
- Lazy horizontal staggered grids have the same behavior on the horizontal axis with **items of different widths**

Summary

- Use the `verticalScroll` or `horizontalScroll` modifiers to display a small list of composables in a Column or a Row
- For dynamic and larger lists use `LazyColumn` and `LazyRow` for the vertical and horizontal scenarios, respectively
- You can program various interactions with a displayed list/grid including *search*, *sort*, *refresh*, *add*, *update* and *delete*

Resources

- **Lists and grids**

https://developer.android.com/jetpack/compose/lists

- **Codelab - Basic layouts in Compose**

https://developer.android.com/codelabs/jetpack-compose-layouts

- **Lazy layouts in Compose**

https://www.youtube.com/watch?v=1ANt65eoNhQ