Topics:

Async Image

Material 3 example

Lazy Lists

Lazy Column, Lazy Row

Lazy Grid, Lazy Staggered Grid

To use material 3 components, you must add the material 3 dependencies in your **build.gradle(app)** file under dependencies:

```
implementation("androidx.compose.material3:material3:1.1.2")
    implementation("androidx.compose.material3:material3-window-size-class:1.1.2")
    implementation("androidx.compose.material3:material3-adaptive:1.0.0-alpha05")
    implementation("androidx.compose.material3:material3-adaptive-navigation-suite:1.0.0-alpha02")
```

To use the images from the internet you have to use the third-party dependency called ***Coil***

```
implementation("io.coil-kt:coil-compose:2.5.0")
```

After adding all the dependencies you **must sync the gradle** file with the project

Using the material3 components:

Example: ElevatedCard

```
ElevatedCard(onClick = { }) {
      Text("Hello Google")
}
```

In the above code snippet we have used the elevated card. We can use another composable inside card ex. Text(), and Image and also it has the onClick property such that if the use click on the card the we can do something ex. Increasing the size of the card and extra content

Using the Image from the internet:

```
AsyncImage(model = "Image Url", contentDescription = "null")
```

In the above code snippet we have used the AsyncImage composable which comes from the coil library.

Which takes the image url and the content description.

Using the Lazy Column

```
LazyColumn(modifier = Modifier
    .fillMaxSize()
    .background(Color.LightGray)
    .padding(5.dp),
    verticalArrangement = Arrangement.spacedBy(10.dp),
    horizontalAlignment = Alignment.CenterHorizontally )
{
      item{
            Text(text = "GDSC WCE")
      }
}
```

In the above code snippet we have used the lazy column and in the scope of the lazy column we have used the item lambda function to use the single-item

And to show the scrollable list of the items we have to use the items function and pass the list or count of the items to the items function and use that particular composable inside the scope of the items function.

```
items (items = data.names)
{
    name -> MyCard(name = name)

}}
```

for example we have passed the list of names to the items function and called the MyCard composable and passed the single item to the MyCard function

And also we can modify the LazyColumn by the modifier.

Same we can use for the Row.

Also the important properties related to the lazy column and the lazy row is that the arrangement and the alignment

Arrangement is the way in which the things in the column or row will be placed.

Alignment is the alignment of every item in the list from its cross axis

Main axis: The axis along which the items are placed.

Cross axis: The axis perpendicular to the main axis

	Main	Cross
Column	Vertical	Horizontal
Row	Horizontal	Vertical

Arrangement is along the main axis.

Alignment is along the cross axis.

Grid View:

Vertical Grid: LazyVerticalGrid

Horizontal Grid: LazyHorizontalGrid

```
LazyVerticalGrid(columns = GridCells.Fixed(2)){
    items(count = 10){
        AsyncImage(model = "", contentDescription = "")
    }
}
```

In the above code snippet we have used the Lazy Vertical Grid in which we have passed the no of columns and used items function to show the multiple items given the count of 10, and used the Async Image means it will generate 10 image grid having the columns 2;

Columns:

Columns	
Fixed	No of columns are fixed
Adaptive	No of columns can be varied according the width of the component.

```
LazyVerticalGrid(columns = GridCells.Adaptive(100.dp)){
    items(count = 10){
        AsyncImage(model = "", contentDescription = "")
    }
}
```

In the Grid Cells Adaptive we have passed the min size of the image as 100.dp

Staggered Grid View:

Vertical : LazyStaggeredVerticalGrid

Horizontal : LazyStaggeredHorizintalGrid

```
LazyVerticalStaggeredGrid(columns = StaggeredGridCells.Adaptive(150.dp))
{
    items(count = 10){
        AsyncImage(model = "", contentDescription = "")
    }
}
```

Same we can use fixed and adaptive columns here.