**Grid View Drag & Drop**

In grid view drag and drop, widgets can be drag and dropped in a grid format without overlapping with each other within a Box (Container with fixed size).

# **Widget Tree Structure**

* Scaffold
  + App Bar (With fixed Size)
  + Stack
    - Container (Seat Type Container with fixed size)
    - Container (Seat Container with Fixed Size and Margins)
      * Single Child Scroll View (Get Scroll offset)
        + Sized Box (Dynamic Height >= Blue Container Height)

Stack

Grid View

Container (Having Borders)

Stack

Positioned (Multiple Numbers)

Long Press Draggable

Multiple positioned widgets are stored inside the stack widget with different x and y coordinates. Whenever the draggable containers are dragged their x and y coordinates are updated.

Note: - Whenever the draggable widget is dragged, it will take the offset (x and y coordinates) according to the whole screen size even if it is inside a container less than the size of the screen. But whenever the dragged widget is placed inside the fixed size container than the position of the widget will be according to the size of the container. So that spacing around the container should be removed from the offset of the dragged widget and scroll amount should be added in order to place the widget at exact place where it is dropped.

# **Requirements**

1. Cross Axis Count (Dynamic Integer)
2. Grid Gap (Constant Integer according to the screen size)
3. Container Size (Seat type container = Grid Gap \* 2 or 3)
4. Margin (horizontal, top = remaining width after the grid has occupied width)
5. Margin Button (remaining space after the grid has occupied the height)
6. Grid Height (The height of the grid which could be scrolled and be used to display the seats)
7. Main Axis Count (Dynamic Integer)
8. Scroll Controller (To get the amount scrolling done)
9. App Bar Height (If app bar is available)
10. List of seat types
11. List of seats

These variables are most required and will be used to display the Grid View with correct alignment. The other variables that will be required along with these variables to calculate the position of the dragged widget are screen height and width, grid width, height and width of seats.

Note: - Height and width of the seats are dynamic. That means the size of seats are different.

# **Adding widget and updating position**

While dragging and dropping the widget, the offset given by the draggable widget is modified and added inside the list of the seats.

1. New widget will be created only If the dragged widget is dropped inside the grid container
2. The minimum position of x and y coordinates is 0, maximum position of x coordinate is less than the grid width (Grid width – widget width), and maximum position of y coordinate is less than grid height (Grid height – widget height) so that the dragged widget stay inside the container
3. If scrolling is done then the scroll amount is added to the y coordinate
4. The new coordinates are then checked for alignment within the grid lines. If the widget position lies between the grid gap, then it is aligned to the nearest grid lines. While updating the position, the difference between previous and new offset is calculated and then checked.
5. Then the modified offset is checked with another widget’s offset to find out whether it touches other widget available inside the grid or not. “for” loop is used to check with every widget inside the list of seats. While updating the position, the widget is not checked with itself.
6. If the dragged widget reaches the end of grid container, then new lines equal to the size of the dragged widget is added at the bottom of the grid container and it will be scrollable
7. Then new seats are saved again

# **Showing previous data (also inside different screen size)**

If previous data is available, then the data are shown inside the grid view. If the data are being showed in different screen, then the grid view is scaled according to the screen width but the main axis count modified If previous main axis count is greater.

1. The grid gap, main axis count, cross axis count, and seats are saved
2. If the saved grid gap or main axis count or cross axis count does not match with the current ones, then the dimensions and coordinates of the previous seats are modified
3. If previous main or cross axis count is not equal to current ones, then the previous ones are assigned to the current ones.
4. Then the seats coordinates and sizes are also changed according to the new grid dimensions
5. New offset = (previous offset / previous grid gap) \* new grid gap
6. New Size = (previous size / previous grid gap) \* new grid gap
7. Then new dimensions with new seat coordinates are saved

# **Updating widget property and size**

The size of each widget can be changed and the cross-axis count can be also modified.

1. The widget with new height and width is checked for overlapping with other widgets inside the grid area, if the new size off the widget overlaps with other than its size is not updated.
2. If the overlapping does not occur, then the size is updated without exceeding the grid area.
3. Then the property of the widget is also updated and seats are saved again.

# **Updating cross axis count**

When the cross-axis count is updated, the whole dimensions are changed.

1. Previous dimensions and seat data are retrieved.
2. The new cross-axis count assigned to the current one, and the grid gap and other variables are calculated again.
3. The seats’ coordinates and size are also changed according to the new dimensions of the grid.
4. The new grid dimension and seat data are saved.