RIVET

Pivotal Insights | Focused Solutions

Mobile Internship 2023

Document Number	MBL_IT_001_V.1.0
Owner	Mobile Domain
Classification	Medium
Type	Intern's Training
Access Permission	Mobile

Contents

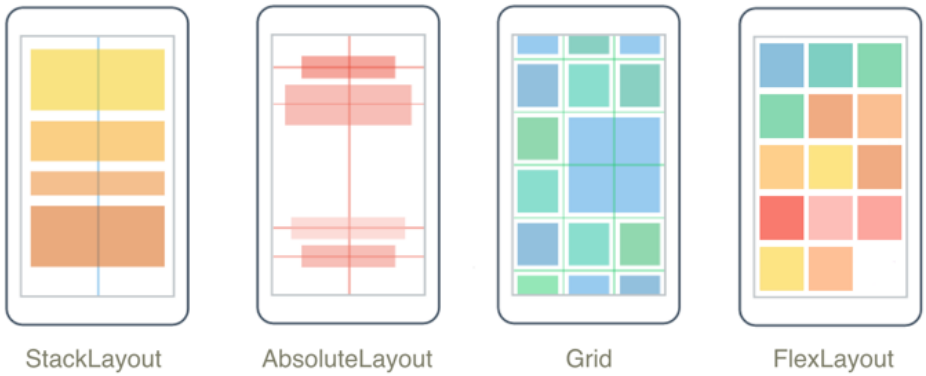
1 CHAPTER 1: LAYOUTS AND PROPERTIES 2

LAYOUT 2
StackLayout 2
Horizontal StackLayout 3
Vertical StackLayout 3
Absolute Layout 3
Grid 4
FlexLayout 4
PROPERTIES 5
HeightRequest 5
WidthRequest 5
HorizontalOptions 5
VerticalOptions 6
Margin 7
Padding 7
Orientation 8
Spacing 9
LayoutBounds 10
LayoutFlags 11
Column & Row 12
ColumnDefinitions & RowDefinitions 12
ColumnSpan & RowSpan 13
ColumnSpacing & RowSpacing 13
Direction 15
AlignItems 15
JustifyContent 16
Order 17
Basis 18
Grow 19
AlignSelf 19
EXERCISES 21

Chapter 1: Layouts and Properties

Layout

.NET Multi-platform App UI (.NET MAUI) layout classes allow you to arrange and group UI controls in your application. Choosing a layout class requires knowledge of how the layout positions its child elements, and how the layout sizes its child elements.



Types of Layouts

StackLayout

StackLayout organizes elements in a one-dimensional stack, either horizontally or vertically. The Orientation property specifies the direction of the elements, and the default orientation is Vertical.

Syntax:

```
<StackLayout>
 …

</StackLayout>
```

Horizontal StackLayout

HorizontalStackLayout organizes child views in a one-dimensional horizontal stack and is a more performant alternative to a StackLayout.

Syntax:

```
<HorizontalStackLayout>
    ...
</HorizontalStackLayout>
```

Vertical StackLayout

VerticalStackLayout organizes child views in a one-dimensional vertical stack and is a more performant alternative to a StackLayout.

Syntax:

```
<VerticalStackLayout>
    ...
</VerticalStackLayout>
```

Absolute Layout

AbsoluteLayout is used to position and size children using explicit values. The position is specified by the upper-left corner of the child relative to the upper-left corner of the AbsoluteLayout, in device-independent units. AbsoluteLayout also implements a proportional positioning and sizing feature.

Syntax:

```
<AbsoluteLayout>
    ...
</AbsoluteLayout>
```

Grid

Grid is a layout that organizes its children into rows and columns, which can have proportional or absolute sizes. By default, a Grid contains one row and one column. In addition, a Grid can be used as a parent layout that contains other child layouts.

Syntax:

```
<Grid>
   ...

</Grid>
```

FlexLayout

FlexLayout is a layout that can arrange its children horizontally and vertically in a stack and can also wrap its children if there are too many to fit in a single row or column. In addition, FlexLayout can control orientation and alignment, and adapt to different screen sizes.

Syntax:

```
<FlexLayout>
      ...

</FlexLayout>
```

Properties

HeightRequest

Gets or sets the desired height of view.

Parent Control

View

Syntax

```
<BoxView
      HeightRequest="200"/>
```

WidthRequest

Gets or sets the desired width of view.

Parent Control

View

Syntax

```
<BoxView
      WidthRequest="200"/>
```

HorizontalOptions

Gets or sets the LayoutOptions that define how the element gets laid in a layout.

Parent Control

View

Syntax

```
<BoxView
     HorizontalOptions="Start"/>
```

Values

Center
Start
End
Fill

Note: *The default value of this property is Center.*

VerticalOptions

Gets or sets the LayoutOptions that define how the element gets laid in a layout

Parent Control

View

Syntax

```
<BoxView
        VerticalOptions="Start"/>
```

Values

Center
Start
End
Fill

Note: *The default value of this property is Center.*

Margin

The Margin property represents the distance between an element and its adjacent elements, and is used to control the element's rendering position, and the rendering position of its neighbors. Margin values can be specified on layouts and views.

Parent Control

View

Syntax

```
<BoxView
    Margin="16, 8, 16, 8"/>
```

Padding

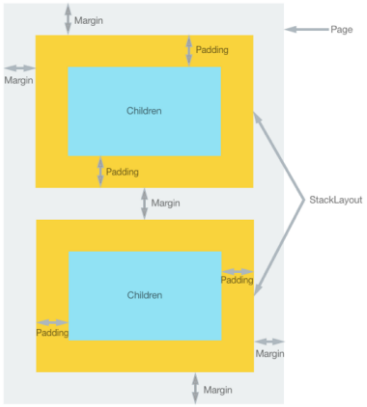
The Padding property represents the distance between an element and its child elements and is used to separate the control from its own content. Padding values can be specified on pages, layouts, and views.

Parent Control

View

Syntax

```
<BoxView
    Padding="16, 8, 16, 8"/>
```



Orientation

Represents the direction such as Horizontal & Vertical in which child views are positioned.

Parent Control

StackLayout

Syntax

```
<StackLayout
  Orientation="Vertical">
  ...
</StackLayout>
```

Values

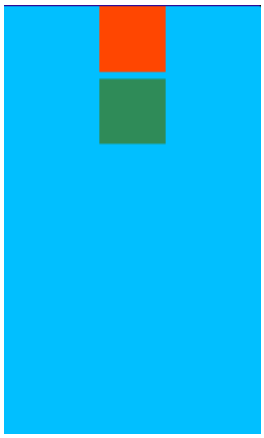
Horizontal

Vertical

Note: *The default value of this property is Vertical.*

Example

```
<StackLayout
  Orientation="Vertical"
  Spacing="10"
  Background="DeepSkyBlue">

  <BoxView
    HeightRequest="100"
    Color="OrangeRed"
    WidthRequest="100"/>

  <BoxView
    HeightRequest="100"
    Color="SeaGreen"
    WidthRequest="100"/>

</StackLayout>
```



Spacing

Spacing, of type double, indicates the amount of space between each child view.

Parent Control

StackLayout, HorizontalStackLayout, VerticalStackLayout,

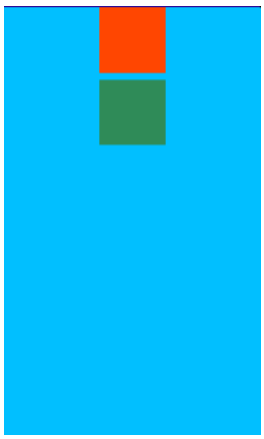
Syntax

```
<StackLayout
      Spacing="10">
  …
</StackLayout>
```

Note: *The default value of this property is 0.*

Example

```
<StackLayout
  Orientation="Vertical"
  Spacing="10"
  Background="DeepSkyBlue">

  <BoxView
    HeightRequest="100"
    Color="OrangeRed"
    WidthRequest="100"/>

  <BoxView
    HeightRequest="100"
    Color="SeaGreen"
    WidthRequest="100"/>

</StackLayout>
```



LayoutBounds

LayoutBounds, of type Rect, which is an attached property that represents the position and size of a child.

Parent Control

AbsoluteLayout,

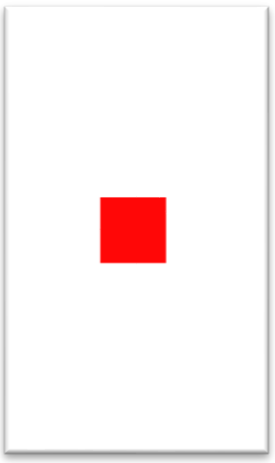
Syntax

```
<AbsoluteLayout>

      <BoxView
            AbsoluteLayout.LayoutBounds="0.5,0.5,110,25"/>
</AbsoluteLayout>
```

Note: The default value of this property is (0,0,AutoSize,AutoSize).

Example

```
<AbsoluteLayout>

  <BoxView
      Color="Red"
      AbsoluteLayout.LayoutBounds="0.5,0.5,100,100"
      AbsoluteLayout.LayoutFlags="PositionProportional" />

</AbsoluteLayout>
```



LayoutFlags

LayoutFlags, of type AbsoluteLayoutFlags, which is an attached property that indicates whether properties of the layout bounds used to position and size the child are interpreted proportionally.

Parent Control

AbsoluteLayout,

Syntax

```
<AbsoluteLayout>

      <BoxView
            AbsoluteLayout.LayoutFlags="PositionProportional"/>
</AbsoluteLayout>
```

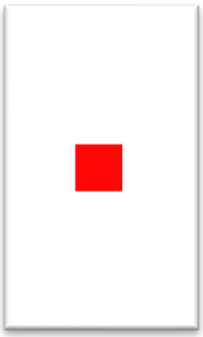
Values

All
HeightProportional
WidthProportional
None
PositionProportional
SizeProportional
XProportional
YProportional

Note: The default value of this property is AbsoluteLayoutFlags.None.

Example

```
<AbsoluteLayout>

   <BoxView
        Color="Red"
        AbsoluteLayout.LayoutBounds="0.5,0.5,100,100"
        AbsoluteLayout.LayoutFlags="PositionProportional" />

 </AbsoluteLayout>
```



Column & Row

Column, of type int, which is an attached property that indicates the column alignment of a view within a parent Grid.

Row, of type int, which is an attached property that indicates the row alignment of a view within a parent Grid.

Parent Control

Grid

Syntax

```
<Grid>
  <BoxView
    Grid.Column="1"
    Grid.Row="1" />
</ Grid>
```

Note: The default value of this property is 0.

ColumnDefinitions & RowDefinitions

ColumnDefinitions, of type ColumnDefinitionCollection, is a list of ColumnDefinition objects that define the width of the grid columns.

RowDefinitions, of type RowDefinitionCollection, is a list of RowDefintion objects that define the height of the grid rows.

Parent Control

Grid

Syntax

```
<Grid RowDefinitions="*, *, *, 100"
      ColumnDefinitions="150,*"

</Grid>
```

ColumnSpan & RowSpan

ColumnSpan, of type int, which is an attached property that indicates the total number of columns that a view spans within a parent Grid.

RowSpan, of type int, which is an attached property that indicates the total number of rows that a view spans within a parent Grid.

Parent Control

Grid

Syntax

```
<BoxView
    Grid.ColumnSpan="2"/>


 <BoxView
     Grid.RowSpan="2"/>
```

Note: The default value of this property is 1.

ColumnSpacing & RowSpacing

ColumnSpacing, of type double, indicates the distance between grid columns.

RowSpacing, of type double, indicates the distance between grid rows.

Parent Control

Grid

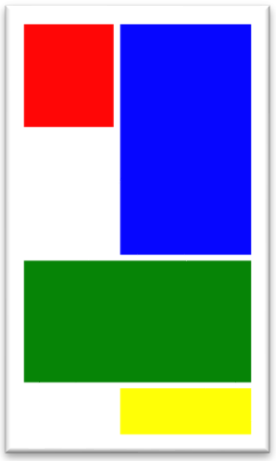
Syntax

```
<Grid
   RowSpacing="10"
   ColumnSpacing="10">
```

Note: The default value of this property is 0.

Example

```
<Grid RowDefinitions="*, *, *, 100"
          ColumnDefinitions="150,*"
          RowSpacing="10"
          ColumnSpacing="10"
          Margin="10">

        <BoxView
            Color="Red"
            Grid.Row="0"/>

        <BoxView
            Color="Blue"
            Grid.RowSpan="2"
            Grid.Column="1"/>

        <BoxView
            Color="Green"
            Grid.Row="2"
            Grid.ColumnSpan="2"/>

        <BoxView
            Color="Yellow"
            Grid.Row="3"
            Grid.Column="2"/>
</Grid>
```



Direction

Setting the property to Column causes the children of the FlexLayout to be arranged in a single column of items.

Parent Control

FlexLayout

Syntax

```
<FlexLayout
        Direction="Column">
</FlexLayout>
```

Values

Column
ColumnReverse
Row
RowReverse

Note: The default value of this property is Row.

AlignItems

The AlignItems property is of type FlexAlignItems and specifies how items are aligned on the cross axis.

Parent Control

FlexLayout

Syntax

```
<FlexLayout
        AlignItems="Center">
</FlexLayout>
```

Values

Center
Start
Stretch
End

Note: The default value of this property is Stretch.

JustifyContent

The JustifyContent property is of type FlexJustify, and specifies how items are arranged on the main axis.

Parent Control

FlexLayout

Syntax

```
<FlexLayout
        JustifyContent="SpaceEvenly">
</FlexLayout>
```

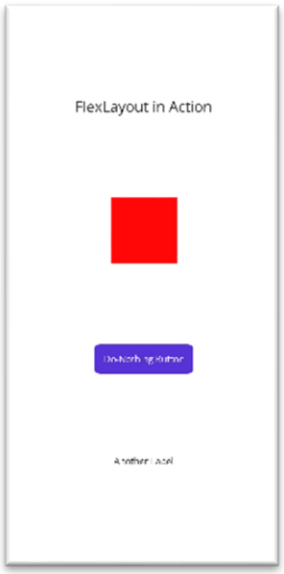
Values

Center
Start
End
SpaceAround
SpaceBetween
SpaceEvenly

Note: The default value of this property is Start.

Example

```
<ContentPage
xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
xmlns:x=http://schemas.microsoft.com/winfx/2009/xaml
x:Class="MauiDemos.Views.User_Interface.FlexPage"
Title="FlexPage">

    <FlexLayout Direction="Column"
                AlignItems="Center"
                JustifyContent="SpaceEvenly">

        <Label Text="FlexLayout in Action"
               FontSize="Large" />

        <BoxView
            HeightRequest="100"
            WidthRequest="100"
            Color="Red"/>

        <Button Text="Do-Nothing Button" />

        <Label Text="Another Label" />

    </FlexLayout>

</ContentPage>
```



Order

The order property to change the layout order.

Parent Control

FlexLayout

Syntax

```
<FlexLayout
       <BoxView
            FlexLayout.Order="1"/>
</FlexLayout>
```

Note: The default value of this property is 0.

Basis

Basis to specify a pixel size or a percentage that indicates how much space the item occupies on the main axis.

Parent Control

FlexLayout

Syntax

```
<FlexLayout
      <BoxView
            FlexLayout.Basis="50"/>
</FlexLayout>
```

Note: The default value of this property is Auto.

Grow

Grow, of type float, which is an attached property that specifies the amount of available space the child should use on the main axis.

Parent Control

FlexLayout

Syntax

```
<FlexLayout
      FlexLayout.Grow="1">
</FlexLayout>
```

Note: The default value of this property is 0.

AlignSelf

AlignSelf, of type FlexAlignSelf, which is an attached property that indicates how the layout engine will distribute space between and around children for a specific child along the cross axis.

Parent Control

FlexLayout

Syntax

```
<FlexLayout
      FlexLayout.AlignSelf="Center">
</FlexLayout>
```

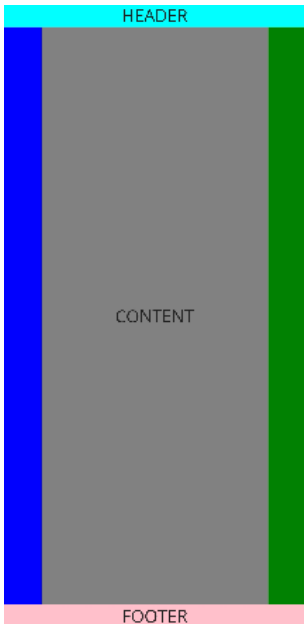
Values

Stretch
Start
Auto
Center
End

Note: The default value of this property is Auto.

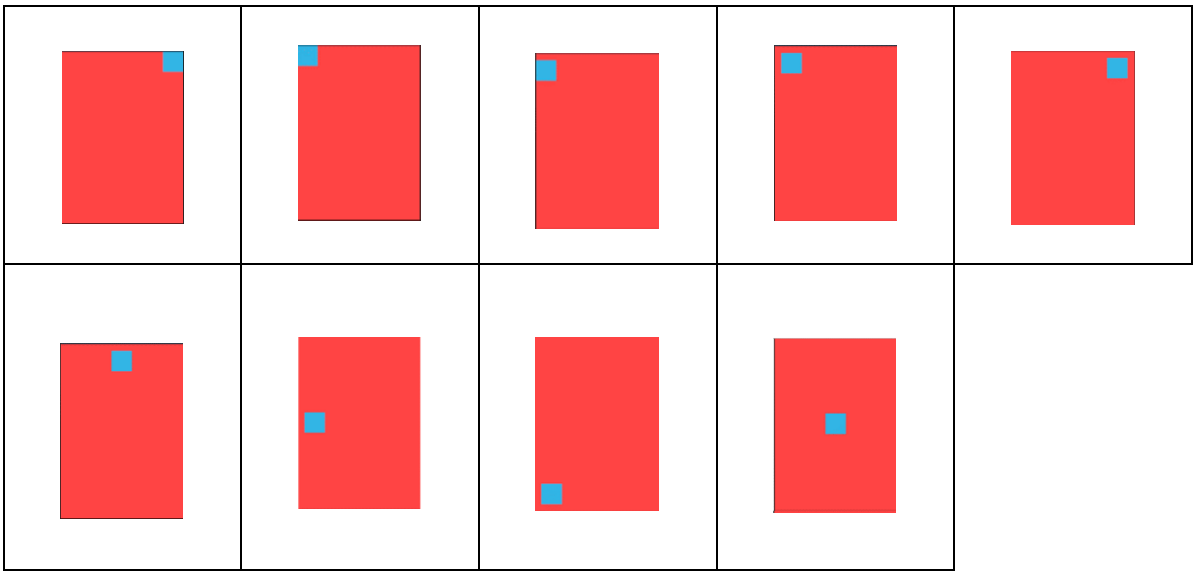
Example

```
<FlexLayout Direction="Column">

        <!-- Header -->
        <Label Text="HEADER"
               FontSize="Large"
               FlexLayout.AlignSelf="Stretch"
               BackgroundColor="Aqua"
               HorizontalTextAlignment="Center" />

        <!-- Body -->
        <FlexLayout FlexLayout.Grow="1">

            <!-- Content -->
            <Label Text="CONTENT"
                   FontSize="Large"
                   BackgroundColor="Gray"
                   HorizontalTextAlignment="Center"
                   VerticalTextAlignment="Center"
                   FlexLayout.Grow="1" />

            <!-- Navigation items-->
            <BoxView FlexLayout.Basis="50"
                     FlexLayout.Order="-1"
                     Color="Blue" />

            <!-- Aside items -->
            <BoxView FlexLayout.Basis="50"
                     Color="Green" />

        </FlexLayout>

        <!-- Footer -->
        <Label Text="FOOTER"
               FontSize="Large"
               BackgroundColor="Pink"
               HorizontalTextAlignment="Center" />
    </FlexLayout>
```

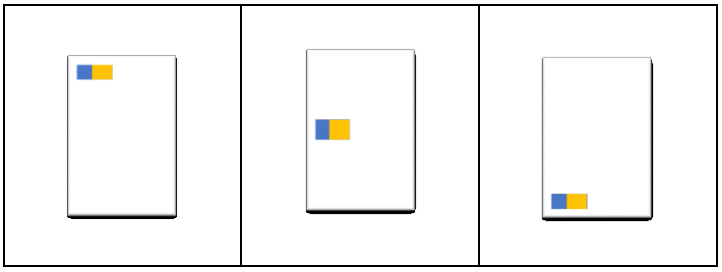


Exercises

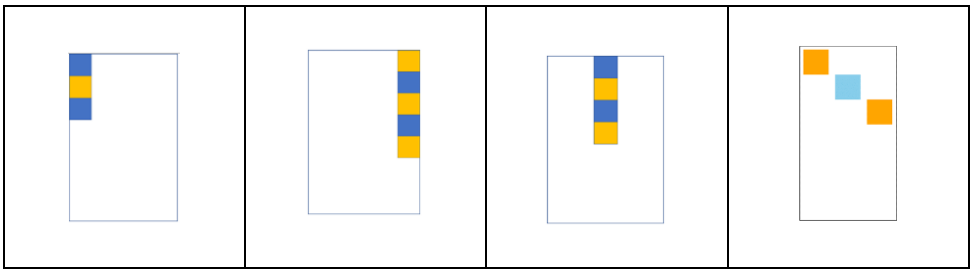
1. Create the below design using **StackLayout**.



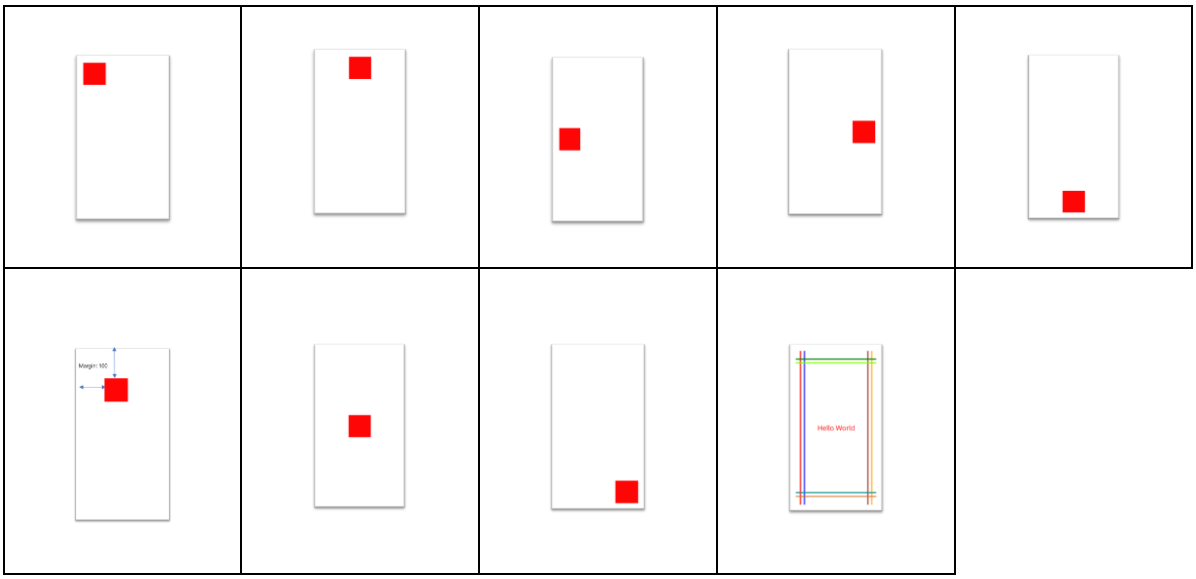
2. Create the below design using **HorizontalStackLayout.**



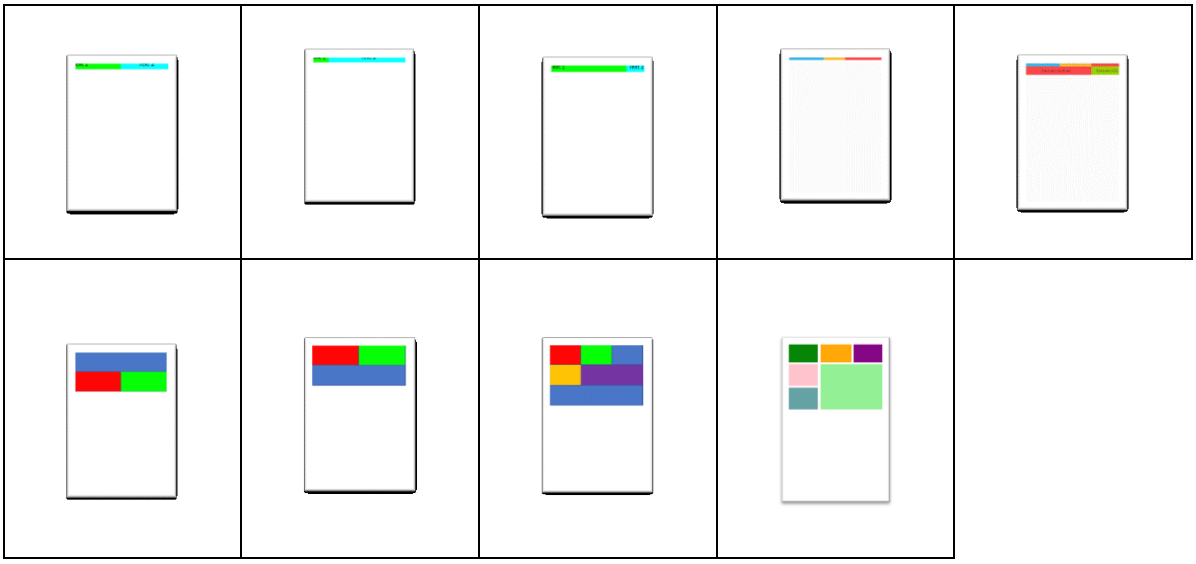
3. Create the below design using **VerticalStackLayout.**



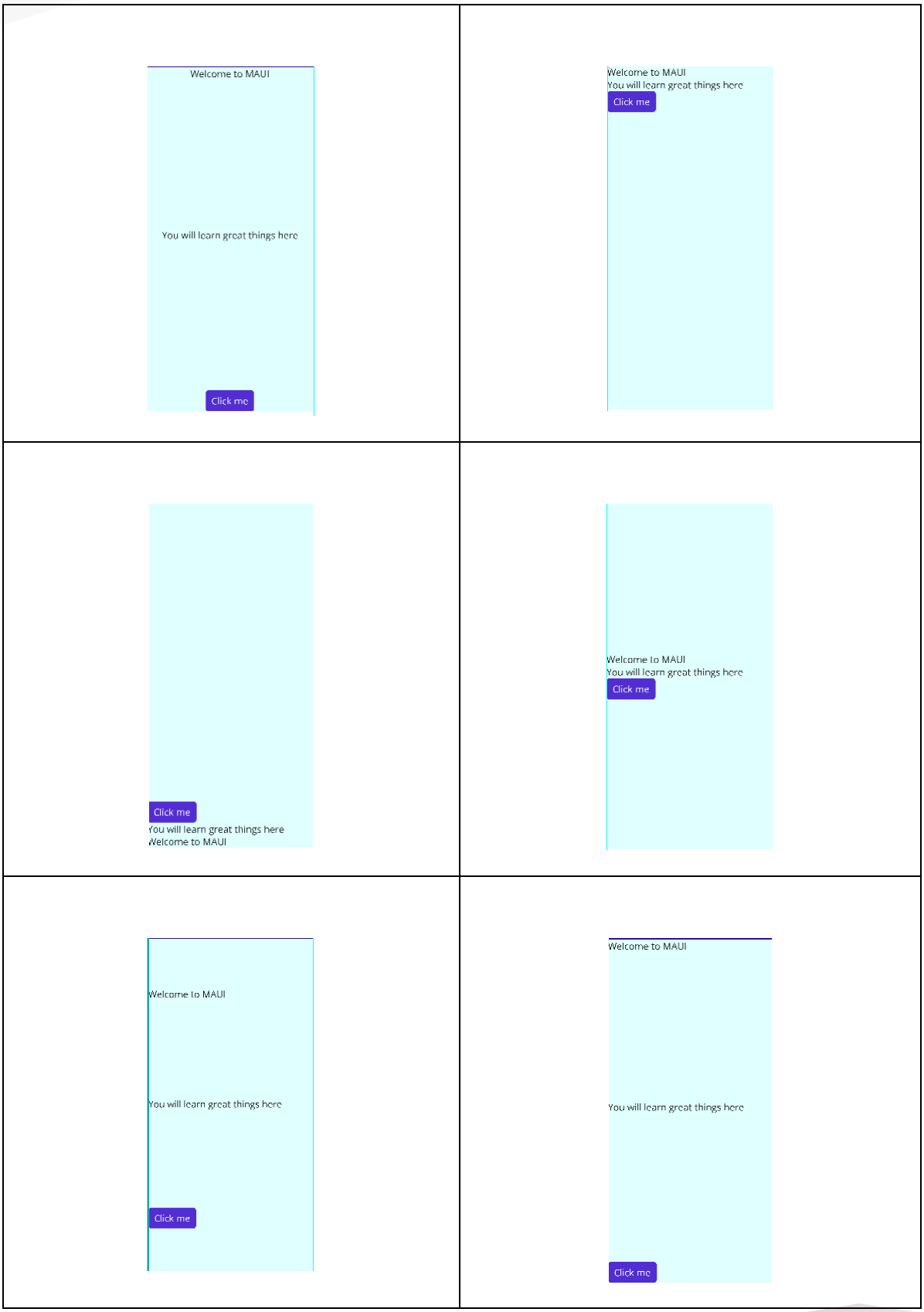
4. Create the below design using **AbsoluteLayout.**



5. Create the below design using **Grid.**



6. Create the below design using **FlexLayout.**



7. Create the below designs using any Layout.