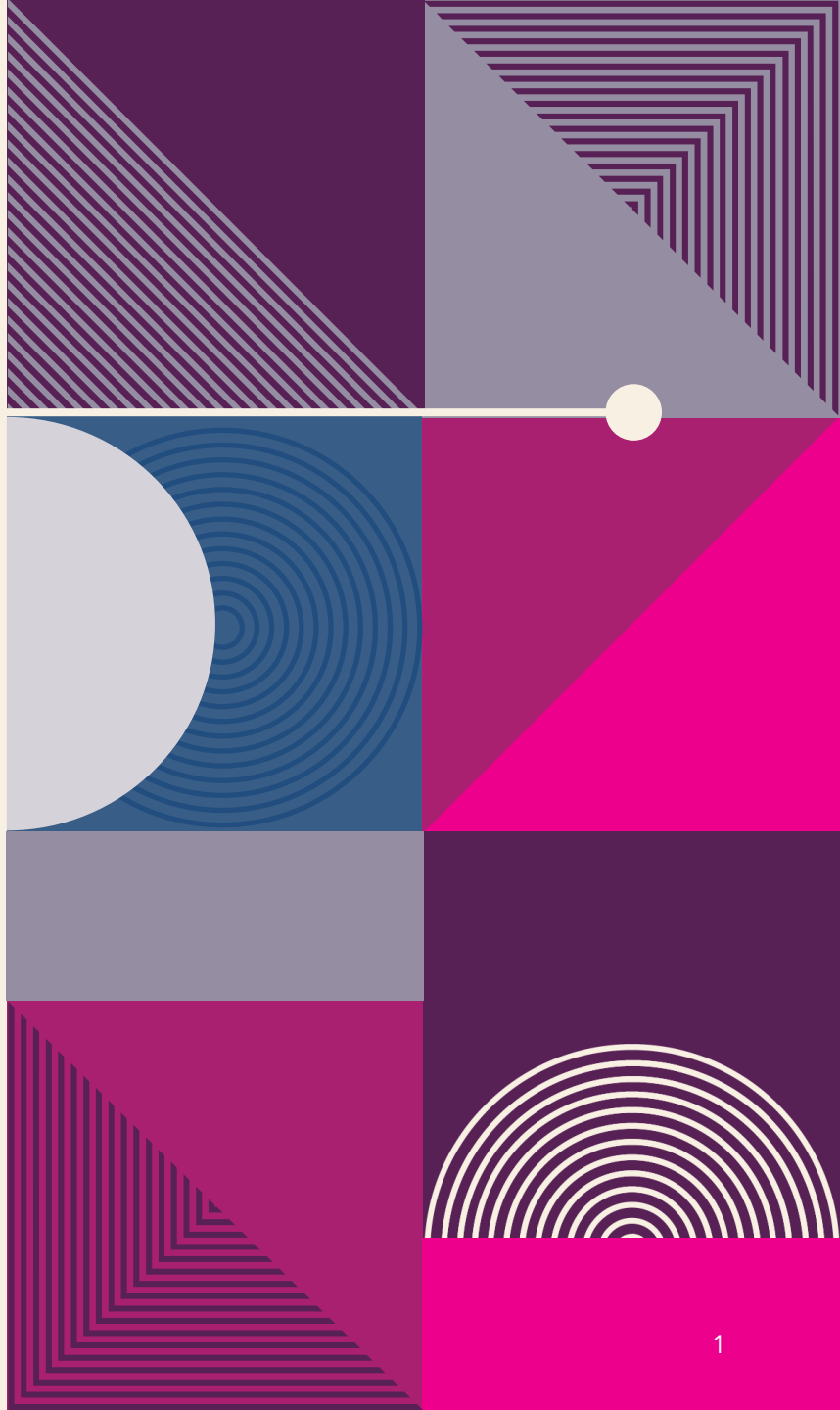
Applying Interactivity



SETTING UP GESTUREDETECTOR : THE BASICS

- The **GestureDetector** widget detects gestures such as **tap**, **double tap**, **long press**, **pan**, **vertical drag**, **horizontal drag**, and **scale**.
- It has an optional **child** property, and if a child widget is specified, the gestures apply only to the child widget. If the child widget is omitted, then the **GestureDetector** fills the entire parent instead.
- If you need to catch **vertical drag** and **horizontal drag** at the same time, use the **pan gesture**.
- If you need to catch a **single-axis drag**, then use either the **vertical drag** or **horizontal drag** gesture.

The following are the **GestureDetector** gestures that you can listen for and take appropriate action:

- **Tap**
  - onTapDown
  - onTapUp
  - onTap
  - onTapCancel
- **Double tap**
  - onDoubleTap
- **Long press**
  - onLongPress
- **Pan**
  - onPanStart
  - onPanUpdate
  - onPanEnd

- **Vertical drag**
  - onVerticalDragStart
  - onVerticalDragUpdate
  - onVerticalDragEnd

- **Horizontal drag**
  - onHorizontalDragStart
  - onHorizontalDragUpdate
  - onHorizontalDragEnd
- **Scale**
  - onScaleStart
  - onScaleUpdate
  - onScaleEnd

Creating the Gesture, Drag-and-Drop App

1. Create a new Flutter project. For this project, you need to create only the pages folder.
2. Open the main.dart file. Change the primarySwatch property from blue to lightGreen.
3. Open the home.dart file and add to the body a SafeArea with a SingleChildScrollView as a child.

– Add a Column as a child of the SingleChildScrollView.
– For the Column children list of Widget, add a method call to _buildGestureDetector(), _buildDraggable(), and _buildDragTarget() with Divider widgets between them.

```
body: SafeArea(
  child: SingleChildScrollView(
    child: Column(
      children: <Widget>[
        _buildGestureDetector(),
        Divider(
          color: Colors.black,
          height: 44.0,
        ),
        _buildDraggable(),
        Divider(
          height: 40.0,
        ),
        _buildDragTarget(),
        Divider(
          color: Colors.black,
        ),
      ],
    ),
  ),
),
```

4. Add the **_buildGestureDetector**() **GestureDetector method** after the Widget **build(BuildContext context)** {...}.

5. Return a **GestureDetector** listening to the **onTap**, **onDoubleTap**, **onLongPress**, and **onPanUpdate** gestures.

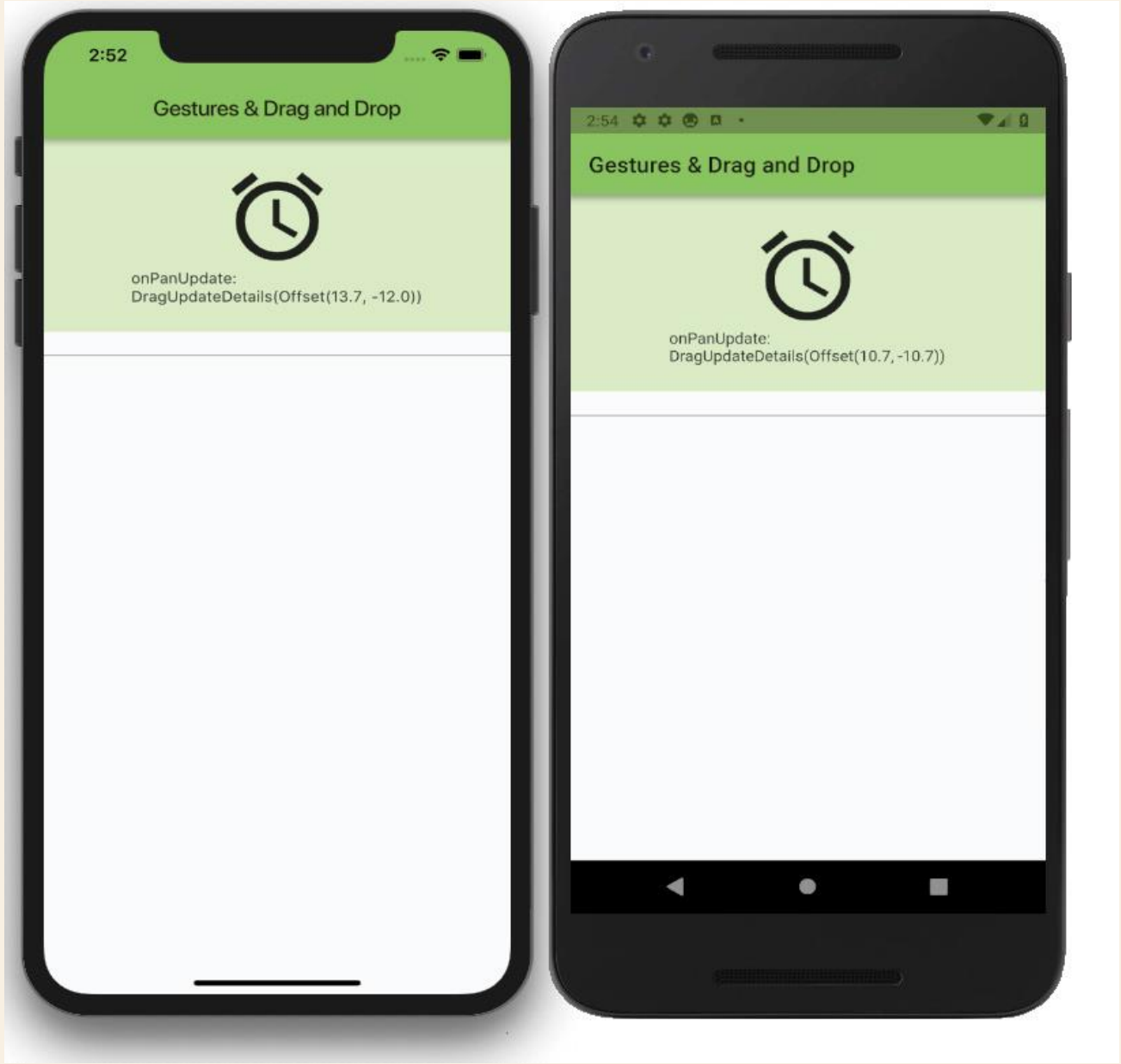
6. To view **captured gestures**, **add** a **Container** as a **child** of the **GestureDetector**.

- The **Container child** is a **Column** displaying an **Icon** and a **Text** widget showing the **gesture detected and pointer location on the screen**.
- **add** the **onVerticalDragUpdate** and **onHorizontalDragUpdate** gestures (properties) but commented them out for you to experiment.

7. To update the screen with the pointer location and to have code reuse,

- **create** the **_displayGestureDetected**(String gesture) method.
- Each gesture passes the String representation of the gesture.
- The **onPanUpdate**, **onVerticalDragUpdate**, and **onHorizontalDragUpdate** gestures (properties) listen to **DragUpdateDetails**.

```
GestureDetector _buildGestureDetector() {
  return GestureDetector(
    onTap: () {
      print('onTap');
      _displayGestureDetected('onTap');
    },
    onDoubleTap: () {
      print('onDoubleTap');
      _displayGestureDetected('onDoubleTap');
    },
    onLongPress: () {
      print('onLongPress');
      _displayGestureDetected('onLongPress');
    },
    onPanUpdate: (DragUpdateDetails details) {
      print('onPanUpdate: $details');
      _displayGestureDetected('onPanUpdate:\n$details');
    },
      child: Container(
      color: Colors.lightGreen.shade100,
      width: double.infinity,
      padding: EdgeInsets.all(24.0),
      child: Column(
        children: <Widget>[
          Icon(
            Icons.access_alarm,
            size: 98.0,
          ),
          Text('$_gestureDetected'),        ],      ),    ),  );}
void _displayGestureDetected(String gesture) {
  setState(() {
    _gestureDetected = gesture;  });}
```


2:52
Gestures & Drag and Drop
onPanUpdate:
DragUpdateDetails(Offset(13.7, -12.0))
2:54
Gestures & Drag and Drop
onPanUpdate:
DragUpdateDetails(Offset(10.7, -10.7))

Gestures: Adding Drag and Drop

Continuing with the previous gestures project, let's add the Draggable and DragTarget methods

8. Create the _buildDraggable() method, which returns a Draggable integer.

- The Draggable child is a Column with the children list of Widget consisting of an Icon and a Text widget.
- The feedback property is an Icon, and the data property passes the Color as an integer value.

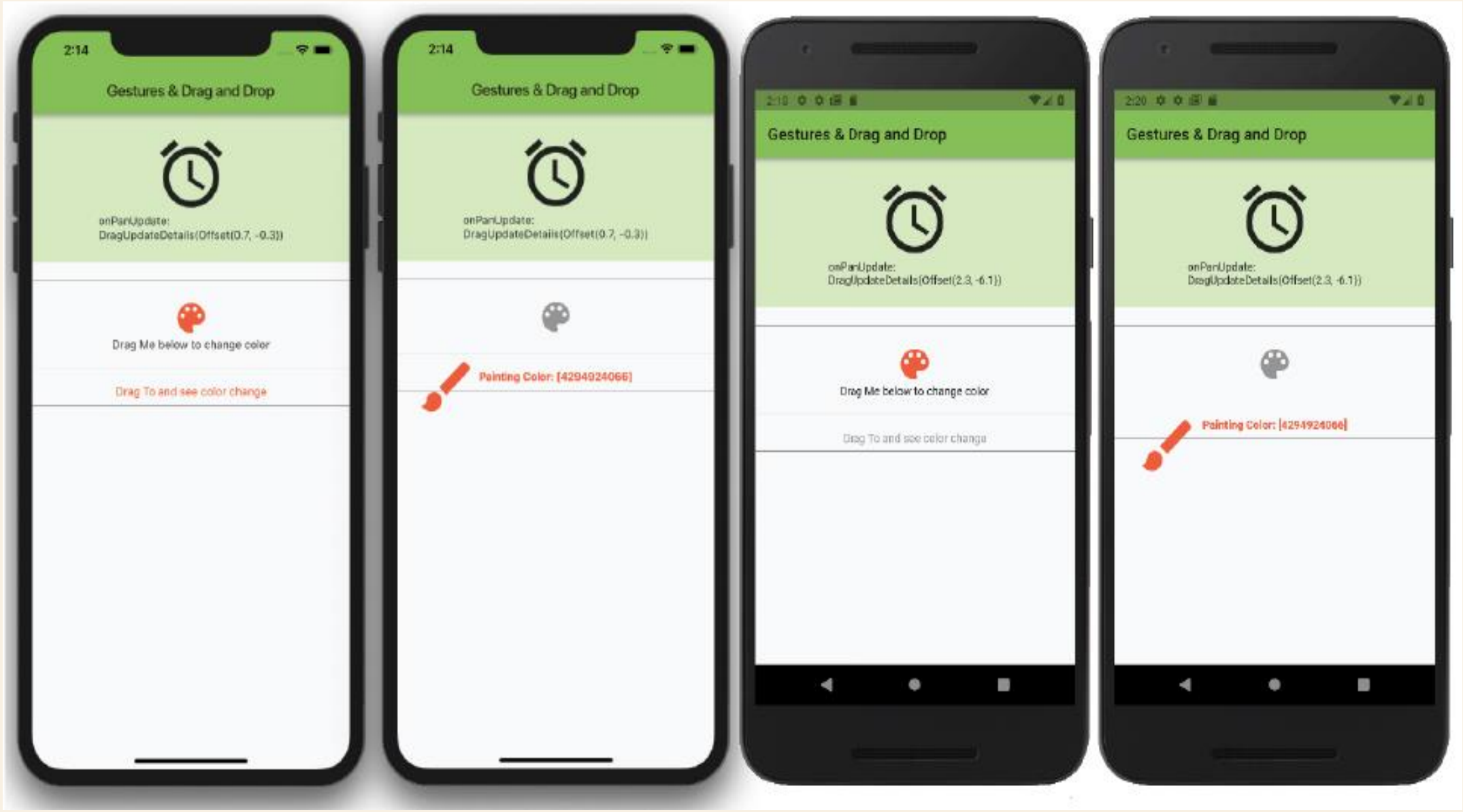
```
Draggable<int> _buildDraggable() {
  return Draggable(
    child: Column(
      children: <Widget>[
        Icon(
          Icons.palette,
          color: Colors.deepOrange,
          size: 48.0,
        ),
        Text(
          'Drag Me below to change color',         ),       ],    ),
    childWhenDragging: Icon(
      Icons.palette,
      color: Colors.grey,
      size: 48.0,
    ),
    feedback: Icon(
      Icons.brush,
      color: Colors.deepOrange,
      size: 80.0,
    ),
    data: Colors.deepOrange.value,
  );
}
```

9. Create the **_buildDragTarget**() method, which returns a **DragTarget** integer.

- To accept **data**, set the **DragTarget onAccept** property value to **colorValue**
- set the **_paintedColor** variable to the **Color(colorValue).**
- The **Color(colorValue)** constructs (**converts**) the **integer value** to a **color**.

10. Set the **builder** property to accept three parameters: **BuildContext**, **List<dynamic> acceptedData**, and **List<dynamic>** of **rejectedData**.

```
DragTarget<int> _buildDragTarget() {
    return DragTarget<int>(
      onAccept: (colorValue) {
        _paintedColor = Color(colorValue);
      },
      builder: (BuildContext context, List<dynamic> acceptedData,
List<dynamic> rejectedData) => acceptedData.isEmpty
          ? Text(
        'Drag To and see color change',
        style: TextStyle(color: _paintedColor),
      )
          : Text(
        'Painting Color: $acceptedData',
        style: TextStyle(
          color: Color(acceptedData[0]),
          fontWeight: FontWeight.bold,
        ),
      ),
    );
  }
}
```


2:14
Gestures & Drag and Drop
onPanUpdate:
DragUpdateDetails(Offset(0.7, -0.3))
Drag Me below to change color
Drag To and see color change
2:14
Gestures & Drag and Drop
onPanUpdate:
DragUpdateDetails(Offset(0.7, -0.3))
Painting Color: [4294924066]
Gestures & Drag and Drop
onPanUpdate:
DragUpdateDetails(Offset(2.3, -6.1))
Drag Me below to change color
Drag To and see color change
2:20
Gestures & Drag and Drop
onPanUpdate:
DragUpdateDetails(Offset(2.3, -6.1))
Painting Color: [4294924066]

USING THE DISMISSIBLE WIDGET

The **Dismissible** widget is dismissed by a dragging gesture. The direction of the drag can be changed by using **DismissDirection** for the direction property.

- **DismissDirection** Dismiss Options

DIRECTION	DISMISSED WHEN. . .
startToEnd	Dragging left to right.*
endToStart	Dragging right to left.*
Horizontal	Dragging either left or right.
Up	Dragging up.
Down	Dragging down.
Vertical	Dragging either up or down.
*** Assuming reading direction is left to right; when reading direction is right to left, these work the opposite ways.**	

Creating the Dismissible App

1. **Create** a **new Flutter project**. For this project, you only need to **create** the **pages** and **classes** folders. **Create** the **Home** Class as a **StatefulWidget**.
2. **Open** the **home.dart** file and **add** to the **body a ListView.builder().**

```
body: ListView.builder(),
```

3. **Add** to the top of the file the import **trip.dart** package that you'll create next.

```
import 'package:flutter/material.dart';
import 'package:ch11_dismissible/classes/trip.dart';
```

4. **Create** a **new Dart file** under the **classes folder**. Right-click the **classes** folder, select **New** ⇨ **Dart** File, enter **trip.dart**, and click the **OK** button to save.

5. Create the **Trip Class**. The **Trip Class** holds the vacation details with an **id, tripName**, and **tripLocation** String variables. **Create** the **Trip** constructor with named parameters by entering the variable names **this.id**, **this.tripName**, and **this.tripLocation** inside the curly brackets ({}).

```
class Trip {
  String id;
  String tripName;
  String tripLocation;

  Trip({this.id, this.tripName, this.tripLocation});
}
```

1. **Edit** the **home.dart** file and after the class **_HomeState extends State<Home>** and before **@override**, **add** the **List** variable **_trips** initialized by an empty **Trip List**..

```
List _trips = [];
```

7. Override the initState() to initialize the _trips List. add items to the _trips List.

```
@override
void initState() {
  super.initState();
  _trips..add(Trip(id: '0', tripName: 'Rome', tripLocation: 'Italy'))
    ..add(Trip(id: '1', tripName: 'Paris', tripLocation: 'France'))
    ..add(Trip(id: '2', tripName: 'Cancun', tripLocation: 'Mexico'))
    ..add(Trip(id: '3', tripName: 'London', tripLocation: 'England'))
    ..add(Trip(id: '4', tripName: 'Sydney', tripLocation: 'Australia'))
    ..add(Trip(id: '5', tripName: 'Miami', tripLocation: 'USA - Florida'))
    ..add(Trip(id: '6', tripName: 'Rio de Janeiro', tripLocation: 'Brazil'))
    ..add(Trip(id: '7', tripName: 'Cusco', tripLocation: 'Peru'))
    ..add(Trip(id: '8', tripName: 'New Delhi', tripLocation: 'India'))
    ..add(Trip(id: '9', tripName: 'Tokyo', tripLocation: 'Japan'));
}
```

8. Create two methods that simulate marking a **Trip** item **completed** or **deleted** in the database. Create the **_markTripCompleted()** and **_deleteTrip()** methods that act as placeholders to write to a database.

```
void _markTripCompleted() {
  // Mark trip completed in Database or web service
}
void _deleteTrip() {
  // Delete trip from Database or web service
}
```

9. Set the **ListView.builder** constructor with the **itemCount** argument set to **_trips.length**, which is the number of rows in the **_trips List**. For the **itemBuilder** argument.

```
itemCount: _trips.length,
```

– The **itemBuilder** returns a **Dismissible** with the **key** property as **Key(_trips[index].id).** The **Key** is the identifier for each widget and must be **unique**,

```
key: Key(_trips[index].id),
```

10. The **Dismissible** has a **background (drag left to right)** and the **secondaryBackground** (**drag left to right**) properties.

– Set the **background** property to the **_buildCompleteTrip()** method

Set the **secondaryBackground** to the **_buildRemoveTrip()** method.

```
child: _buildListTile(index),
background: _buildCompleteTrip(),
secondaryBackground: _buildRemoveTrip(),
```

– The **onDismissed** callback (property) is called when the widget is **dismissed**, providing a function to run code by **removing** the dismissed widget item from the **_trips List**.

– The **next** step is to use the **setState** to **remove** the dismissed item from the **_trips List** by using the **_trips.removeAt(index).**

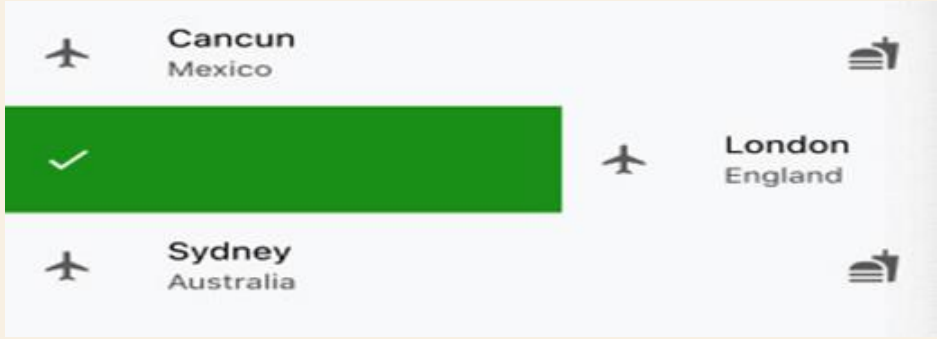
```
body: ListView.builder(
  itemCount: _trips.length,
  itemBuilder: (BuildContext context, int index) {
    return Dismissible(
      key: Key(_trips[index].id),
      child: _buildListTile(index),
      background: _buildCompleteTrip(),
      secondaryBackground: _buildRemoveTrip(),
      onDismissed: (DismissDirection direction) {
        direction == DismissDirection.startToEnd ?
_markTripCompleted() : _deleteTrip();

        // Remove item from List
        setState(() {
          _trips.removeAt(index);
        });
      },
    );
  },),
```

11. Add the **_buildListTile(int index)** Widget method **after** the Widget **build(BuildContextcontext)** {...}. **Return** a **ListTile** and **set** the **title**, **subtitle**, **leading**, and **trailing** properties.

```
ListTile _buildListTile(int index) {
  return ListTile(
          title: Text('${_trips[index].tripName}'),
          subtitle: Text(_trips[index].tripLocation),
          leading: Icon(Icons.flight),
          trailing: Icon(Icons.fastfood),
        );
}
```

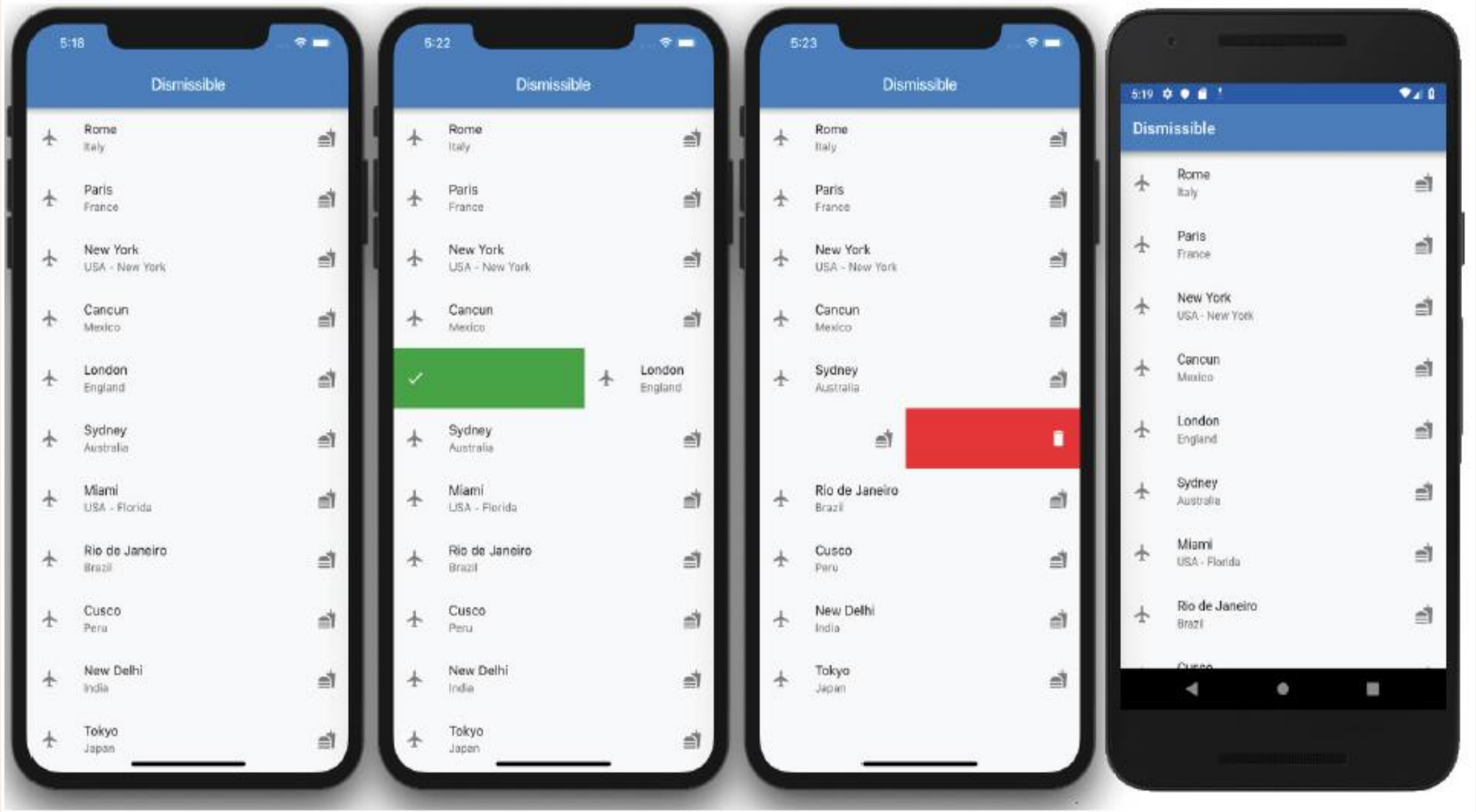
12. Add the **_buildCompleteTrip()** Widget **method** to **return** a **Container** with the **color** as **green** and the child property as a **Padding**. The **Padding child** is a **Row** with the alignment set to **start** (on the left side for left-to-right languages) with a **children** list of **Widget** of an **Icon**.



```
Container _buildCompleteTrip() {
  return Container(
          color: Colors.green,
          child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.start,
              children: <Widget>[
                Icon(
                  Icons.done,
                  color: Colors.white,
                ),
              ],
            ),
          ),
        );
}
```

13. Add the **_buildRemoveTrip()** Widget method to **return** a **Container** with the **color** as **red** and the **child** property as a **Padding**. The **Padding child** is a **Row** with the alignment set to **end** (on the right side for left-to-right languages) with a **children** list of **Widget** of an **Icon**.


```
Container _buildRemoveTrip() {
  return Container(
          color: Colors.red,
          child: Padding(
            padding: const EdgeInsets.all(16.0),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.end,
              children: <Widget>[
                Icon(
                  Icons.delete,
                  color: Colors.white,
                ),
              ],
            ),
          ),      );}
```


5:18
Dismissible
Rome
Italy
Paris
France
New York
USA - New York
Cancun
Mexico
London
England
Sydney
Australia
Miami
USA - Florida
Rio de Janeiro
Brazil
Cusco
Peru
New Delhi
India
Tokyo
Japan
5:22
Dismissible
Rome
Italy
Paris
France
New York
USA - New York
Cancun
Mexico
London
England
Sydney
Australia
Miami
USA - Florida
Rio de Janeiro
Brazil
Cusco
Peru
New Delhi
India
Tokyo
Japan
5:23
Dismissible
Rome
Italy
Paris
France
New York
USA - New York
Cancun
Mexico
Sydney
Australia
Rio de Janeiro
Brazil
Cusco
Peru
New Delhi
India
Tokyo
Japan
5:19
Dismissible
Rome
Italy
Paris
France
New York
USA - New York
Cancun
Mexico
London
England
Sydney
Australia
Miami
USA - Florida
Rio de Janeiro
Brazil