SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER APPLICATIONS

23CAT607- CROSS-PLATFORM APP DEVELOPMENT

I YEAR II SEM

UNIT 3 – INTRODUCTION TO LAYOUTS

TOPIC 3 – Introduction to Gestures, Statement Management in Flutter

INTRODUCTION TO GESTURES

- Gestures are generally defined as any physical action / movement of a user in the intention of activating a specific control of the mobile device.
- Gestures are as simple as tapping the screen of the mobile device to more complex actions used in gaming applications.

Tap − Touching the surface of the device with fingertip for a short period and then releasing the fingertip.

Double Tap − Tapping twice in a short time.

Drag − Touching the surface of the device with fingertip and then moving the fingertip in a steady manner and then finally releasing the fingertip.

Flick − Similar to dragging, but doing it in a speeder way.

Pinch − Pinching the surface of the device using two fingers.

Spread/Zoom − Opposite of pinching.

Panning − Touching the surface of the device with fingertip and moving it in any direction without releasing the fingertip.

Flutter provides an excellent support for all type of gestures through its exclusive widget, **GestureDetector**. GestureDetector is a non-visual widget primarily used for detecting the user's gesture. To identify a gesture targeted on a widget, the widget can be placed inside **GestureDetector** widget. GestureDetector will capture the gesture and dispatch multiple events based on the gesture.

Tap

- **onTapDown**
- **onTapUp**
- **onTap**
- **onTapCancel**

Double tap

- onDoubleTap

Long press

- onLongPress

Vertical drag

- onVerticalDragStart
- onVerticalDragUpdate
- onVerticalDragEnd

Horizontal drag

- onHorizontalDragStart
- onHorizontalDragUpdate
- onHorizontalDragEnd

Pan

- onPanStart
- onPanUpdate
- onPanEnd

let us modify the hello world application to include gesture detection feature and try to understand the concept.

```
body: Center(
   child: GestureDetector(
      onTap: () {
         _showDialog(context);
      },
      child: Text( 'Hello World', )
   )
),
```

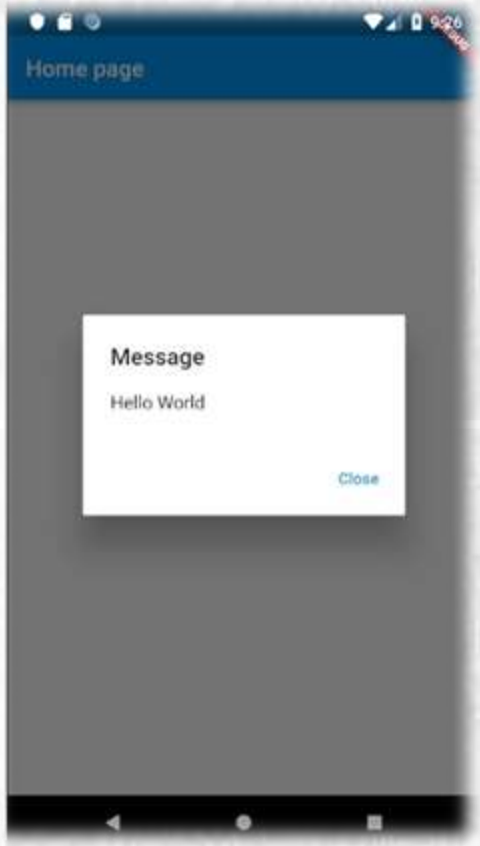
Change the body content of the *MyHomePage* widget

Observe that here we have placed the *GestureDetector* widget above the Text widget in the widget hierarchy, captured the onTap event and then finally shown a dialog window.

Implement the *_showDialog* function to present a dialog when user tabs the hello world message.

It uses the generic showDialog and AlertDialog widget to create a new dialog widget.

```
// user defined function void _showDialog(BuildContext context) {
   // flutter defined function
   showDialog(
      context: context, builder: (BuildContext context) {
         // return object of type Dialog
         return AlertDialog(
            title: new Text("Message"),
            content: new Text("Hello World"),
            actions: <Widget>[
               new FlatButton(
                  child: new Text("Close"),
                  onPressed: () {
                     Navigator.of(context).pop();
                  },
               ),
            ],
         );
      },
   );
}
```



Flutter also provides a low-level gesture detection mechanism through *Listener* widget

- PointerDownEvent
- PointerMoveEvent
- PointerUpEvent
- PointerCancelEvent

Flutter also provides a small set of widgets to do specific as well as advanced gestures.

- **Dismissible** – Supports flick gesture to dismiss the widget.
- **Draggable** – Supports drag gesture to move the widget.
- **LongPressDraggable** – Supports drag gesture to move a widget, when its parent widget is also draggable.
- **DragTarget** – Accepts any *Draggable* widget
- **IgnorePointer** – Hides the widget and its children from the gesture detection process.
- **AbsorbPointer** – Stops the gesture detection process itself and so any overlapping widget also can not able to participate in the gesture detection process and hence, no event is raised.
- **Scrollable** – Support scrolling of the content available inside the widget.

Flutter - State Management

Managing state in an application is one of the most important and necessary process in the life cycle of an application.

A state management can be divided into two categories

Ephemeral

Last for a few seconds like the current state of an animation or a single page like current rating of a product. *Flutter* supports its through StatefulWidget.

app state

Last for entire application like logged in user details, cart information, etc., *Flutter* supports its through scoped_model.

Navigation and Routing

In any application, navigating from one page / screen to another defines the work flow of the application. The way that the navigation of an application is handled is called Routing. Flutter provides a basic routing class – MaterialPageRoute and two methods **- Navigator.push and Navigator.pop**, to define the work flow of an application

MaterialPageRoute

MaterialPageRoute is a widget used to render its UI by replacing the entire screen with a platform specific animation.

```
MaterialPageRoute(builder: (context) => Widget())
```

Navigation.push

Navigation.push is used to navigate to new screen using MaterialPageRoute widget.

```
Navigator.push( context, MaterialPageRoute(builder: (context) => Widget()), );
```

Navigation.pop

Navigation.pop is used to navigate to previous screen.

```
Navigator.push(context);
```

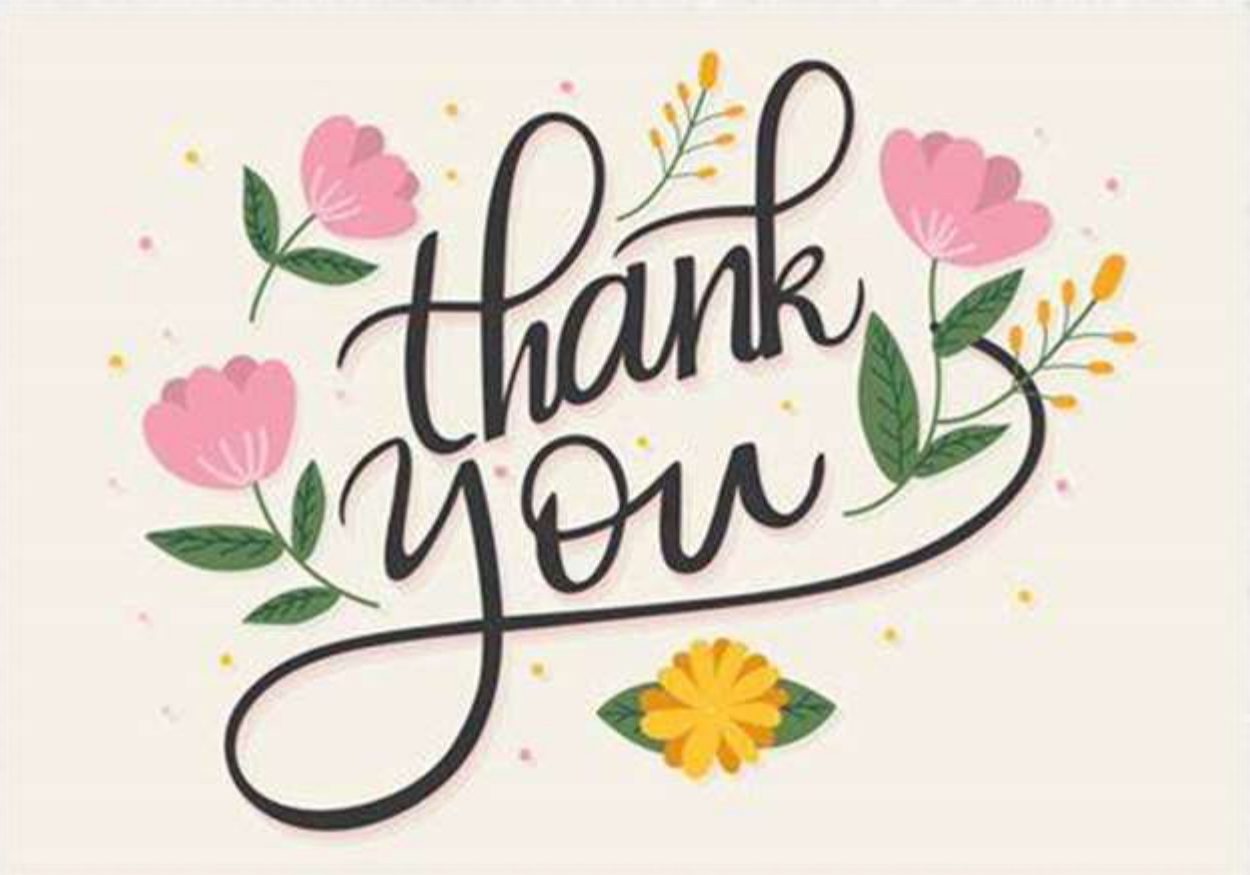
- Create a new Flutter application in Android studio, product_nav_app

Copy the assets folder from ***product_nav_app*** to ***product_state_app*** and add assets inside the ***pubspec.yaml*** file.

```
flutter:
  assets:
  - assets/appimages/floppy.png
  - assets/appimages/iphone.png
  - assets/appimages/laptop.png
  - assets/appimages/pendrive.png
  - assets/appimages/pixel.png
  - assets/appimages/tablet.png
```

•Let us create a Product class to organize the product information.

```
class Product {
    final String name;
    final String description;
    final int price;
    final String image;
    Product(this.name, this.description, this.price, this.image);
}
```


thank
you