Quiz 4

Due Dec 6 at 11:59pm **Points** 30 **Questions** 25 **Time Limit** None
Allowed Attempts 2

Instructions

This quiz covers content covered in lectures 5.0 - 5.2

Questions ***can*** have more than a single correct answer. Please make sure you consider all possible answers when marking your answers.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	43 minutes	18 out of 30

Correct answers are hidden.

Score for this attempt: **18** out of 30
Submitted Dec 5 at 5:05pm
This attempt took 43 minutes.

Partial

Question 1

0.75 / 1 pts

What sensor(s) are available on most smartphones (Select all that apply)?

- [✓] Proximity sensor
- [✓] Bluetooth
- [ ] X-Ray

- ☐ Temperature sensor
- ✓ Image sensor

Partial

Question 2

0.67 / 1 pts

Sensing can be used for ________ (Select all that apply)?

- ✓ Enhance the UX
- ✓ Communication
- ✓ Recognize activities
- ✓ Fall detection

Incorrect

Question 3

0 / 1 pts

The following is true about fall detection using sensors (Select all that apply).

- ☐ Elderly don't accelerate quickly, so fall detection is easier.
- ✓ All data are from simulated falls.
- ☐ The best place to put the sensor is at the waist.
- ✓ It mostly use accelerometer and gyro.

Partial

Question 4

0.67 / 1 pts

What's true about location sensing (Select all that apply)?

- ✓ It can be used for safety monitoring
- ✓ We can use location to customize UX.
- ✓ Location-aware apps are difficult to build because you need to use multiple widgets.
- ✓ If used incorrectly, it can violate user trust.

Question 5

1 / 1 pts

What's true about the following statements (Select all that apply)?

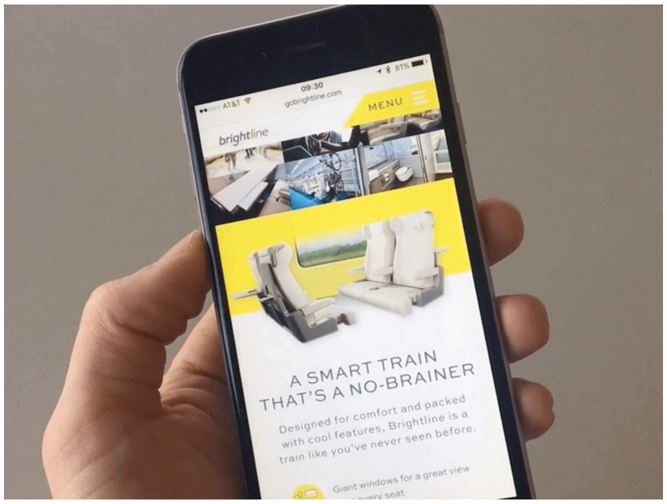
- ✓ Your application should allow users say no when asking for their location.
- ✓ Your application can ask for user's location with a reason.
- ☐ Your application should store location logs for optimization.
- ✓ Points a user has spent more than ThresTime in within ThresDistance of the point called Stay points.

Partial

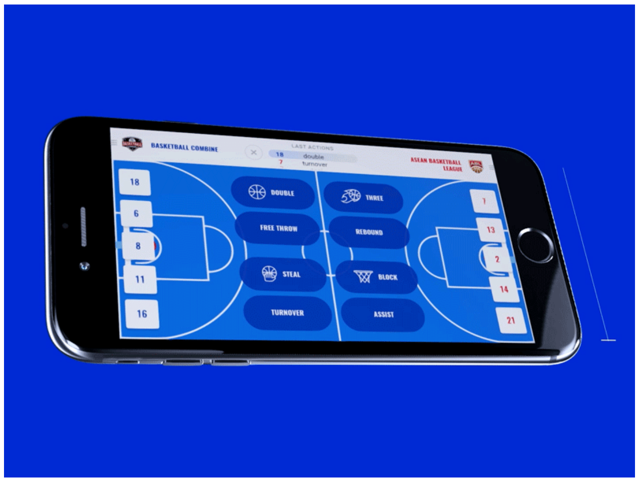
Question 6

0.5 / 1 pts

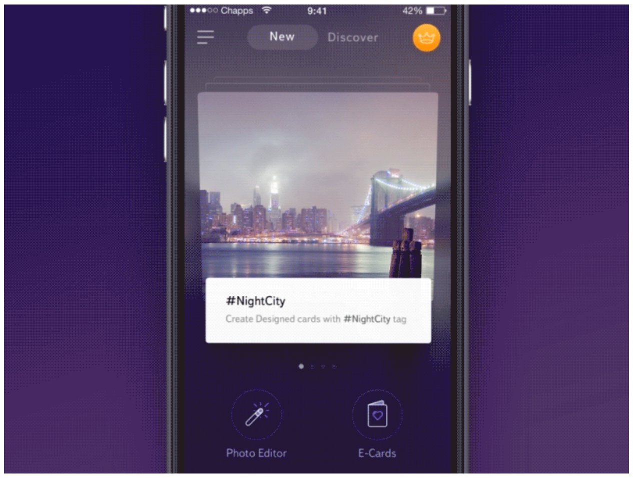
Match the correct use case to its corresponding descriptions.


brightline
MENU
A SMART TRAIN THAT'S A NO-BRAINER
Designed for comfort and packed with cool features, Brightline is a train like you've never seen before.
Giant windows for a great view

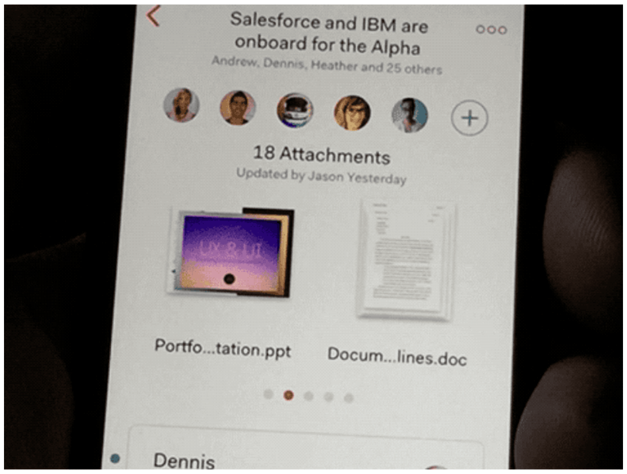
Use case 1



Use case 2



Use case 3


Salesforce and IBM are onboard for the Alpha
Andrew, Dennis, Heather and 25 others
18 Attachments
Updated by Jason Yesterday
Portfo...tation.ppt
Docum...lines.doc
Dennis

Use case 4

Use case 1	Enhance the UX - Rota
Use case 2	Enhance the UX - Char
Use case 3	Enhance the UX - Para
Use case 4	Enhance the UX - Laye

Partial

Question 7

0.5 / 1 pts

Match the figure to its corresponding activity.

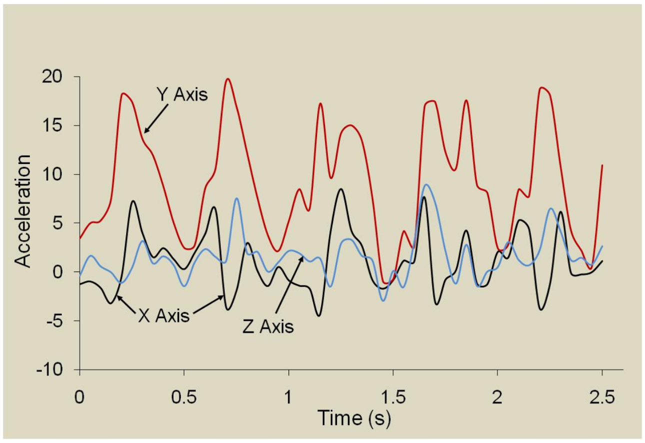


Figure 1

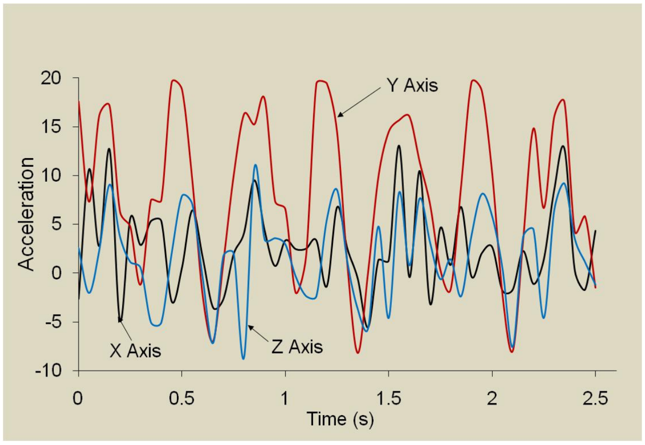


Figure 2

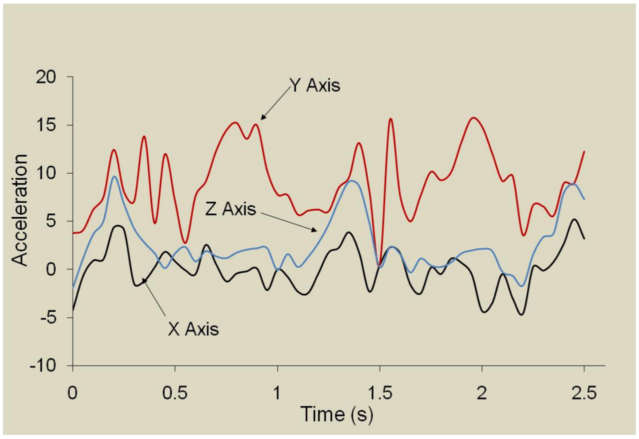


Figure 3

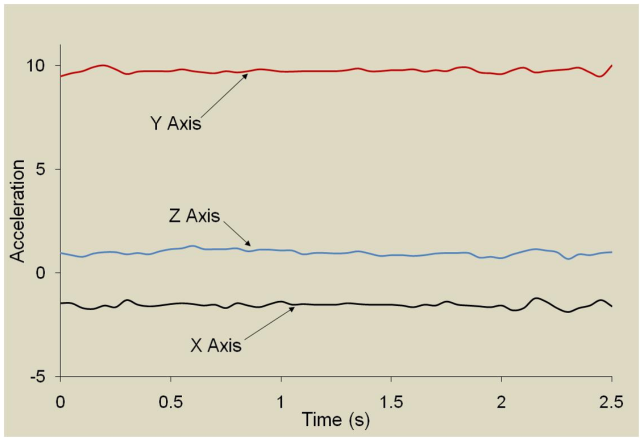


Figure 4

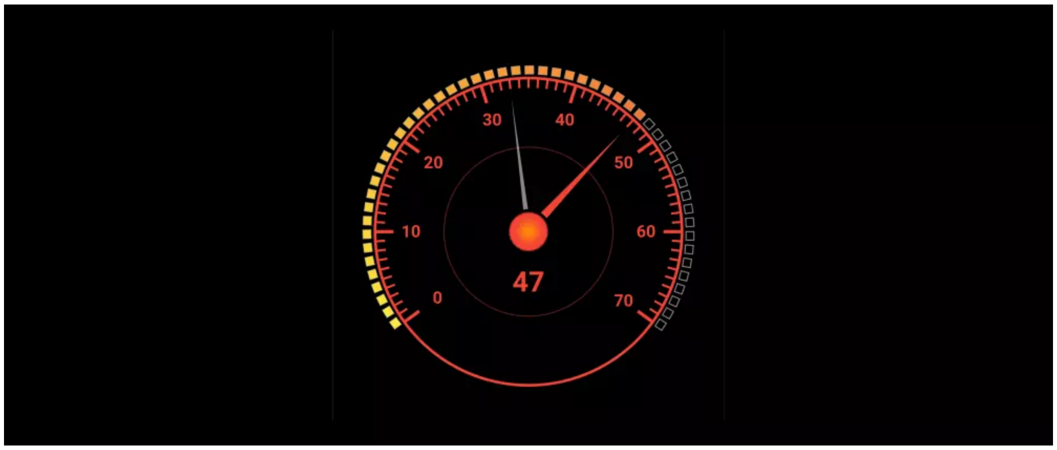
Figure 1	Running
Figure 2	Walking
Figure 3	Climbing stairs
Figure 4	Standing

Partial

Question 8

1.33 / 2 pts

You want to build a smartphone movement speedometer (shown below), read the code carefully, and select the correct statements.



```
import 'dart:math';

import 'package:flutter/material.dart';

import 'package:sensors/sensors.dart';

import 'speedometer.dart';



class SpeedometerContainer extends StatefulWidget {

  @override

  _SpeedometerContainerState createState() => _SpeedometerContainerState();

}



class _SpeedometerContainerState extends State<SpeedometerContainer> {

  double velocity = 0;

  double highestVelocity = 0.0;



  @override

  void initState() {

    userAccelerometerEvents.listen((UserAccelerometerEvent event) {

      _onAccelerate(event);

    });

    super.initState();

  }



  void _onAccelerate(UserAccelerometerEvent event) {

    double newVelocity = sqrt(
      event.x * event.x + event.y * event.y + event.z * event.z
    );


    if ((newVelocity - velocity).abs() < 1) {
      return;
    }


    setState(() {
      velocity = newVelocity;


      if (velocity > highestVelocity) {
        highestVelocity = velocity;
      }
    });
  }


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      body: Stack(
        children: [
          Container(
            padding: EdgeInsets.only(bottom: 64),
            alignment: Alignment.bottomCenter,
            child: Text(
              'Highest speed:\n${highestVelocity.toStringAsFixed(2)} km/h',
              style: TextStyle(
                  color: Colors.white
              ),
              textAlign: TextAlign.center,
            )
          ),
          Center(
            child: Speedometer(
              speed: velocity,
              speedRecord: highestVelocity,
            )
          )
        ]
      )
    );
  }
}
```

```
_drawNeedle(
  0.15 + (speedRecord / 100),
  Colors.white54,
  size.width / 120
);
_drawNeedle(
  0.15 + (speed / 100),
  Colors.red,
  size.width / 70
);
void _drawNeedle(double rotation, Color color, double width) {
  paintObject
    ..style = PaintingStyle.fill
    ..color = color;
  Path needlePath = Path()
    ..moveTo(center.dx - width, center.dy)
    ..lineTo(center.dx + width, center.dy)
    ..lineTo(center.dx, center.dy + size.width / 2.5)
    ..moveTo(center.dx - width, center.dy);
  _drawRotated(rotation, () {
    canvas.drawPath(needlePath, paintObject);
  });
}
```

```
void _drawSpeed() {
  TextSpan span = new TextSpan(
    style: new TextStyle(
      fontWeight: FontWeight.bold,
      color: Colors.red,
      fontSize: size.width / 12
    ),
    text: '${speed.toStringAsFixed(0)}'
  );
  TextPainter textPainter = TextPainter(
    text: span,
    textDirection: TextDirection.ltr,
    textAlign: TextAlign.center
  );
  textPainter.layout();
 final textCenter = Offset(
    center.dx,
    center.dy + (size.width / 10) + (textPainter.width / 2)
  );
 final textTopLeft = Offset(
    textCenter.dx - (textPainter.width / 2),
    textCenter.dy - (textPainter.width / 2)
  );
  textPainter.paint(canvas, textTopLeft);
}
```

[✓] Velocity is represented as x, y and z direction.

✓ This widget wraps the actual speedometer (that has a visual representation on the screen). It captures the current velocity and forwards this value to the widget that is to be created.

✓ UserAccelerometerEvents […] describe the velocity of the device, but don't include gravity. They can also be thought of as just the user's affect on the device.

✓ During initState(), we bind a listener to accelerometerEvents, which is a Stream of events.

Incorrect

Question 9

0 / 2 pts

Read the following code carefully, and select all the options that apply:

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:geolocator/geolocator.dart';

class DashboardScreen extends StatefulWidget {
  @override
  _DashboardState createState() => _DashboardState();
}

class _DashboardState extends State<DashboardScreen> {
  final Geolocator geolocator = Geolocator()..forceAndroidLocationManager;
  Position _currentPosition;
  String _currentAddress;

  @override
  void initState() {
    super.initState();
    _getCurrentLocation();
  }

  _getCurrentLocation() {
    geolocator
        .getCurrentPosition(desiredAccuracy: LocationAccuracy.best)
        .then((Position position) {
      setState(() {
        _currentPosition = position;
      });
      });

      _getAddressFromLatLng();
    }).catchError((e) {
      print(e);
    });
  }

  _getAddressFromLatLng() async {
    try {
      List<Placemark> p = await geolocator.placemarkFromCoordinates(
          _currentPosition.latitude, _currentPosition.longitude);

      Placemark place = p[0];

      setState(() {
        _currentAddress =
            "${place.locality}, ${place.postalCode}, ${place.country}";
      });
    } catch (e) {
      print(e);
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("DASHBOARD"),
      ),
      body: SingleChildScrollView(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Container(
                decoration: BoxDecoration(
                  color: Theme.of(context).canvasColor,
                ),
                padding: EdgeInsets.symmetric(horizontal: 16, vertical: 8),
                child: Column(
                  children: <Widget>[
                    Row(
                      children: <Widget>[
                        Icon(Icons.location_on),
                        SizedBox(
                          width: 8,
                        ),
                        Expanded(
                          child: Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: <Widget>[
                              Text(
                                'Location',
                                style: Theme.of(context).textTheme.caption,
                              ),
                              if (_currentPosition != null &&
                                  _currentAddress != null)
                                Text(_currentAddress,
                                    style:
                                        Theme.of(context).textTheme.bodyText
2),
                            ],
                          ),
                        ),
                        SizedBox(
                          width: 8,
                        ),
                      ],
                    ),
                ],
              ))
          ],
        ),
      ),
    );
  }
}
```

- ✓ The goal of the app is to help users to get all their locations.
- ☐ You need to set android.useAndroidX=true for android project
- ☐ The app will start continuous location updates.
- ✓ _getAddressFromLatLng will be called after _getCurrentLocation.

Partial

Question 10

1.33 / 2 pts

Read the following code carefully, and select all the options that apply:

```
import 'dart:async';
import 'dart:io';

import 'package:camera/camera.dart';
import 'package:flutter/material.dart';

Future<void> main() async {
  // Ensure that plugin services are initialized so that `availableCameras()
`
  // can be called before `runApp()`
  WidgetsFlutterBinding.ensureInitialized();

  // Obtain a list of the available cameras on the device.
  final cameras = await availableCameras();

  // Get a specific camera from the list of available cameras.
  final firstCamera = cameras.first;

  runApp(
    MaterialApp(
      theme: ThemeData.dark(),
      home: TakePictureScreen(
        // Pass the appropriate camera to the TakePictureScreen widget.
        camera: firstCamera,
      ),
    ),
  );
}
```

```
// A screen that allows users to take a picture using a given camera.
class TakePictureScreen extends StatefulWidget {
  const TakePictureScreen({
    super.key,
    required this.camera,
  });

  final CameraDescription camera;

  @override
  TakePictureScreenState createState() => TakePictureScreenState();
}

class TakePictureScreenState extends State<TakePictureScreen> {
  late CameraController _controller;
  late Future<void> _initializeControllerFuture;

  @override
  void initState() {
    super.initState();
    // To display the current output from the Camera,
    // create a CameraController.
    _controller = CameraController(
      // Get a specific camera from the list of available cameras.
      widget.camera,
      // Define the resolution to use.
      ResolutionPreset.medium,
    );

    // Next, initialize the controller. This returns a Future.
    _initializeControllerFuture = _controller.initialize();
  }

  @override
  void dispose() {
    // Dispose of the controller when the widget is disposed.
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Take a picture')),
      // You must wait until the controller is initialized before displaying
the
      // camera preview. Use a FutureBuilder to display a loading spinner un
til the
      // controller has finished initializing.
      body: FutureBuilder<void>(
        future: _initializeControllerFuture,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.done) {
            // If the Future is complete, display the preview.
            return CameraPreview(_controller);
          } else {
            // Otherwise, display a loading indicator.
            return const Center(child: CircularProgressIndicator());
          }
        },
      ),
      floatingActionButton: FloatingActionButton(
        // Provide an onPressed callback.
        onPressed: () async {
          // Take the Picture in a try / catch block. If anything goes wron
g,
          // catch the error.
          try {
            // Ensure that the camera is initialized.
            await _initializeControllerFuture;

            // Attempt to take a picture and get the file `image`
            // where it was saved.
            final image = await _controller.takePicture();

            if (!mounted) return;

            // If the picture was taken, display it on a new screen.
            await Navigator.of(context).push(
              MaterialPageRoute(
                builder: (context) => DisplayPictureScreen(
                  // Pass the automatically generated path to
                  // the DisplayPictureScreen widget.
                  imagePath: image.path,
                ),
              ),
            );
          } catch (e) {
            // If an error occurs, log the error to the console.
            print(e);
          }
        },
        child: const Icon(Icons.camera_alt),
      ),
    );
  }
}

// A widget that displays the picture taken by the user.
class DisplayPictureScreen extends StatelessWidget {
  final String imagePath;

  const DisplayPictureScreen({super.key, required this.imagePath});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Display the Picture')),
      // The image is stored as a file on the device. Use the `Image.file`
      // constructor with the given path to display the image.
      body: Image.file(File(imagePath)),
    );
  }
}
```

✓ In this example, it will create a FloatingActionButton that takes a picture using the CameraController when a user taps on the button.

☐ The code use a CameraPreview to display the camera’s feed.

☐ You can use the CameraController to take pictures using the takePicture() method, which returns an .jpeg file.

✓ After taking the picture, the Image widget will display the saved picture. In this case, the picture is stored as a file on the device.

Incorrect

Question 11 0 / 1 pts

Accessibility equals usability.

- True
- [✓ False]

Question 12 1 / 1 pts

Select all statements that are true about universal design.

- ☐ Discourage conscious action in tasks that require vigilance.
- ✓ Arrange information consistent with its importance.
- ✓ Provide adaptability to the user's pace.
- ✓ Avoid segregating or stigmatizing any users.

Question 13 2 / 2 pts

Match the guideline to its corresponding principle.

Principle 1 - Low physical effort

Principle 2 - Size and Space for Approach and Use

Principle 3 - Tolerance for Error

Principle 4 - Flexibility in Use

Principle 1	Minimize repetitive actic
Principle 2	Provide adequate spac
Principle 3	Provide fail safe feature
Principle 4	Provide choice in meth

Incorrect

Question 14

0 / 1 pts

Select all the statement that are true.

- [ ] Inclusive design is cheap and totally worth it
- [ ] Inclusive design doesn't make your product accessible exclusively for people with different impairments.
- [✓] Implement inclusivity will make your design more accessible, meaning more usable for anyone.

✓ Inclusive design is unifying user experience

Question 15

1 / 1 pts

Select all statements that are true about Unit tests.

✓ A unit test tests a single function, method, or class.

✓ The assert phase verify whether the unit behaves as expected.

✓ Unit tests generally don’t read from or write to disk, render to screen, or receive user actions from outside the process running the test.

☐ The arrange phase is a necessary phase in unit test.

Incorrect

Question 16

0 / 1 pts

Select all statements that are true about Wedget tests.

✓ We use Mockito and Matcher for simulating the interface between software components & Widgets.

☐ The objective of this test is to check whether the widget works and looks true to form.

- [x] Widget testing is performed after unit testing.
- [ ] Widget testing may be performed in isolation with other Widget of the system.

Partial

Question 17

0.5 / 1 pts

Select all statements that are true about Integration tests.

- [ ] Integrations tests will discover all bugs in your software.
- [x] The flutter_driver package will run or "drive" our app for us.
- [x] Integration tests test how pieces of your application work individually.
- [x] With an integration test, one bit of code will run the actual app and another separate bit of code will simulate user interaction.

Partial

Question 18

0.75 / 1.5 pts

Read the following code carefully, and select all the options that are true:

```
class CounterService {
  int _counter = 0;
  int get counter => _counter;

  final int? maxCounterValue;

  CounterService({this.maxCounterValue});

  /// Increases the counter value if [maxCounterValue] has not been reached
yet.
  CounterService inc() {
    if (maxCounterValue == null || _counter < maxCounterValue!) _counter++;

    return this;
  }

  /// Decreases the counter value if [counter] is bigger than zero.
  CounterService dec() {
    if (_counter > 0) _counter--;

    return this;
  }
}
```

```
 test("Verify counter can be increased", () {
    final _sut = CounterService();
    _sut.inc().inc().inc();

    expect(_sut.counter, equals(3));
  });

  test("Verify counter can be increased if below max value", () {
    final _sut = CounterService(maxCounterValue: 2);
    _sut.inc().inc();
    expect(_sut.counter, equals(2));
  });

  test("Verify counter cannot be increased above max value", () {
    final _sut = CounterService(maxCounterValue: 0);

    _sut.inc();
    expect(_sut.counter, equals(0));
  });
```

✓ _sut.inc().inc(); is the “Act” part where code is executed that we want to test.

✓ Make your test independent of other tests.

✓ Verification of results is done by the expect function, which takes the actual value and a Matcher to evaluate the outcome.

Partial

Question 19

1 / 1.5 pts

Read the code below, then select the statements that are true.

```
TextField(
 controller: _controller,
 decoration: InputDecoration(
 border: OutlineInputBorder(),
 labelText: "Enter text"),
),

RaisedButton(
 color: Colors.blueGrey,
 child: Text("Reverse Text"),
 onPressed: () {
 if (_controller.text.isEmpty) return;
 setState(() {
 _reversedText = reverseText(_controller.text);
 });
 },
),

testWidgets("Flutter Widget Test", (WidgetTester tester) async {
 await tester.pumpWidget(MyApp());
 var textField = find.byType(TextField);
 expect(textField, findsOneWidget);
 await tester.enterText(textField, 'Flutter Devs');
 expect(find.text('Flutter Devs'), findsOneWidget);
 print('Flutter Devs');
 var button = find.text("Reverse Text");
 expect(button,findsOneWidget);
 print('Reverse Text');
 await tester.tap(button);
 await tester.pump();
 expect(find.text("sveD rettulF"),findsOneWidget);
 print('sveD rettulF');
});
```

✓ We need to find the text filed and confirm that it exists, find.byType() helps here.

✓ After finding the button, we need to use tester.tap() and pump() to get the reverse text in one application.

✓ Here callback alludes to the testWidgets, a function that permits you to characterize a widget test and makes a widget with the assistance of a WidgetTester.

✓ In this button function, when we add text on the text field and then press the button, the text was reversed shown on our screen.

Incorrect

Question 20

0 / 1 pts

Read the code below, and select the statements that are true.

```
import 'package:flutter_driver/flutter_driver.dart';
import 'package:test/test.dart';
import 'package:tourismandco/models/location.dart';

void main() {
  group('happy path integration tests', () {
    final locations = Location.fetchAll();

    // First, define the Finders. We can use these to locate Widgets from th
e
    // test suite. Note: the Strings provided to the `byValueKey` method mus
t
    // be the same as the Strings we used for the Keys in step 1.
    final locationListItemTextFinder =
        find.byValueKey('location_list_item_${locations.first.id}');

    FlutterDriver driver;

    // Connect to the Flutter driver before running any tests
    setUpAll(() async {
      driver = await FlutterDriver.connect();
    });

    // Close the connection to the driver after the tests have completed
    tearDownAll(() async {
      if (driver != null) {
        driver.close();
      }
    });
    test('a location name appears in location list', () async {
      // Use the `driver.getText` method to verify the counter starts at 0.
      expect(await driver.getText(locationTileOverlayTextFinder), isNotEmpt
y);
    });

    // NOTE one more test to come in the next step!
  });
}
```

☐ In the code, each widget returned by _itemBuilder() is uniquely identifyable in the Flutter widget tree.

☑ Keys can be used to optimize the performance of an app in certain situations, so they are required.

☐ The test will do a basic check to see if the location list is loaded, by finding a widget by the key.

☑ The code will fetch all locations and wait until the Locations screen is loaded.

Question 21

1 / 1 pts

When we go through a unit test case, all we need to know should be inside that test case.

[✓ True]

False

Question 22

1 / 1 pts

A unit test should behave exactly as before wherever or whenever it is been tested without altering the source code.

[✓ True]

False

Question 23

1 / 1 pts

A unit test should focus more on simplicity rather than good coding practices.

[✓ True]

False

Question 24

1 / 1 pts

In unit testing, it is verified that the entire application working as per the specified design.

True

[✓ False]

Question 25

1 / 1 pts

Unit tests should take as much time as needed in order to find all the bugs.

True [✓ False]

Quiz Score: **18** out of 30