Quiz 3

Due Nov 17 at 11:59pm **Points** 30 **Questions** 23 **Time Limit** None
Allowed Attempts 2

Instructions

This quiz covers content covered in lectures 4.0 - 4.2

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	10,352 minutes	18.58 out of 30

Correct answers are hidden.

Score for this attempt: **18.58** out of 30
Submitted Nov 17 at 5:21pm
This attempt took 10,352 minutes.

Partial

Question 1 — 0.75 / 1 pts

The typical form has the following components:

- [x] Structure
- [x] Field labels
- [ ] Logic

- [x] Input fields
- [ ] Assistance

Question 2

1 / 1 pts

The following are true about labels:

- [x] Floating label can help users in filling out the correct field.
- [x] Inline labels is bad when more information is required from the user.
- [ ] It's always good to put labels on top of the input boxes.
- [x] Right-aligned labels can have great completion times, but is less comfortable to look at.
- [x] Placeholder text can increase the chance of error.

Question 3

1 / 1 pts

The following are true about the design of input:

- [x] Avoid generic words such as "submit" for actions when designing buttons.

- [ ] Studies on label placement suggests that forms are completed faster if labels are on left of the fields.

- [x] A good motion gesture improves usability by making applications easier to use in all contexts.

- [x] Interactions become more natural with a gesture-based interface.

- [x] It's better to design for limited interaction

Question 4

1 / 1 pts

Each individual form field should be wrapped in a FormField widget, with the Form widget as a common ancestor of all of those.

- (•) True

- ( ) False

Incorrect

Question 5

0 / 1 pts

The following are true about the design of gestures in UI:

- [x] Well-designed gestures have a shorter learning curve because they are simple and not complexed.

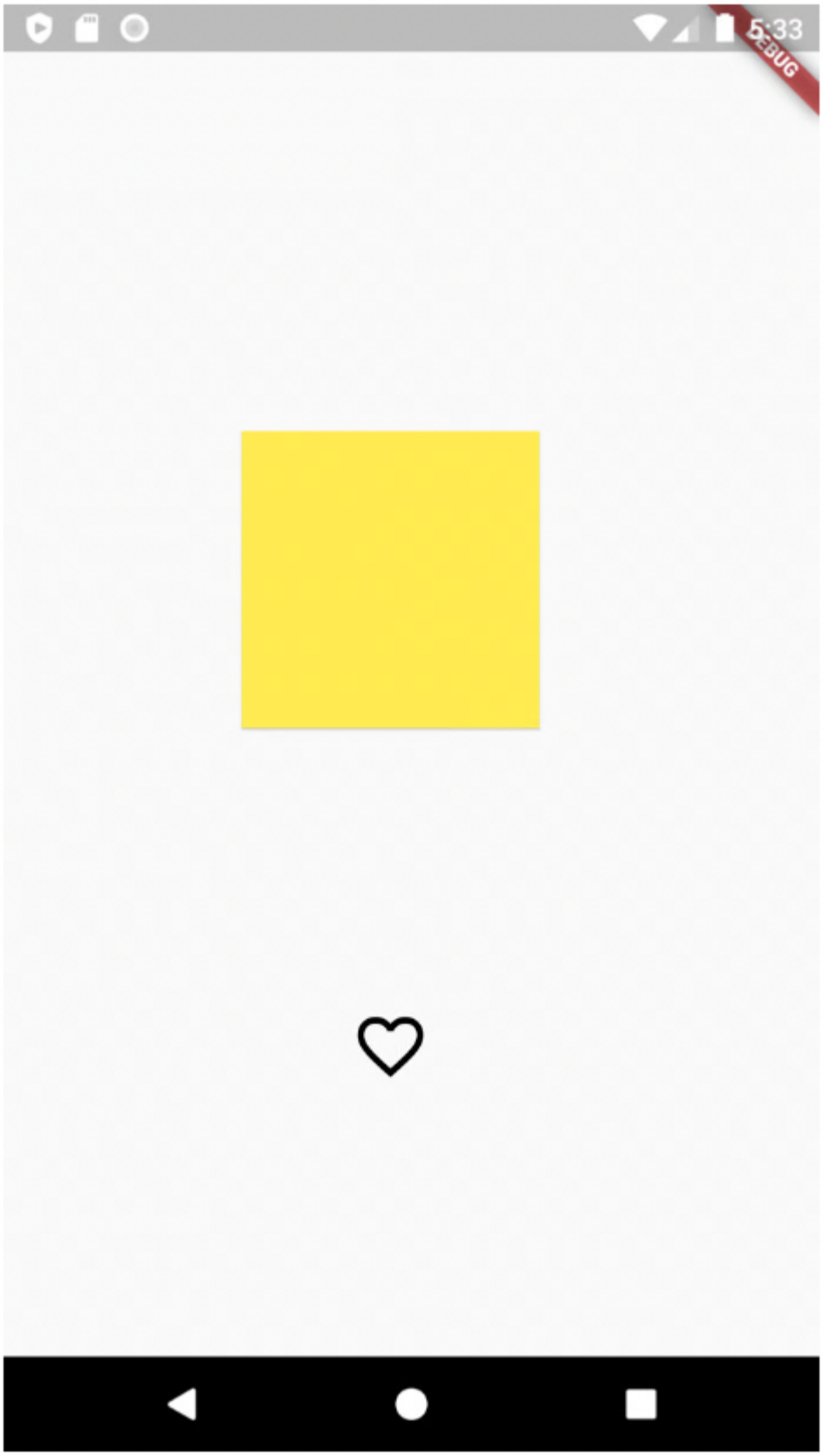
- [x] Interactions are more natural with a gesture-based interface.
- [x] A gesture-based user interface improve task completion rates by making tasks simple and quick.
- [x] When designing gesture-based user interfaces, it's good practice to be creative on all designs.

Partial

Question 6

1 / 2 pts

The following UI has a card widget (the rectangle) and an icon (the heart shape) widget. There are some gestures added on top of the UI, read the code carefully, and answer the following questions. Select all the statements that are true in describing the gestures.




```
class _MyCardWidgetState extends State<MyCardWidget> {
  Color bgColor = Colors.yellow;
  bool makeCircular = false;
  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onLongPress: (){
        setState(() {
```

```
          makeCircular = !makeCircular;
        });
      },
      child:   Card(
        shape: makeCircular? const CircleBorder(): const RoundedRectangleBor
der(),
        child: const SizedBox(
          height: 300,
          width: 300,
        ),
        color: bgColor,
      ),
    );
  }
}
```

```
class _MyCardWidgetState extends State<MyCardWidget> {
  Color bgColor = Colors.yellow;
  bool makeCircular = false;
  double _scaleFactor = 0.4;
  double _baseScaleFactor = 0.5;
  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onScaleStart: (details){
        _baseScaleFactor = _scaleFactor;
      },
      onScaleUpdate: (details){
        setState(() {
          _scaleFactor = _baseScaleFactor * details.scale;
        });
      },
      onScaleEnd: (details){
        // return to initial scale
        _scaleFactor = _baseScaleFactor;
      },
      child:   Transform.scale(
        scale: _scaleFactor,
        child: Card(
          shape: makeCircular? const CircleBorder(): const RoundedRectangleB
orde(),
          child: const SizedBox(
            height: 300,
            width: 300,
          ),
        color: bgColor,
      ),
    );
  }
}
```

- [ ] If user double click one the yellow rectangle, the shape will change.

- [x] The scale gesture will change the scale property based on the value of the _baseScaleFactor

- [ ] If user double click on the yellow rectangle, the shape will change the shape of the card based on the value of the makeCircular property..

- [x] If user long press the yellow rectangle, the shape will change.

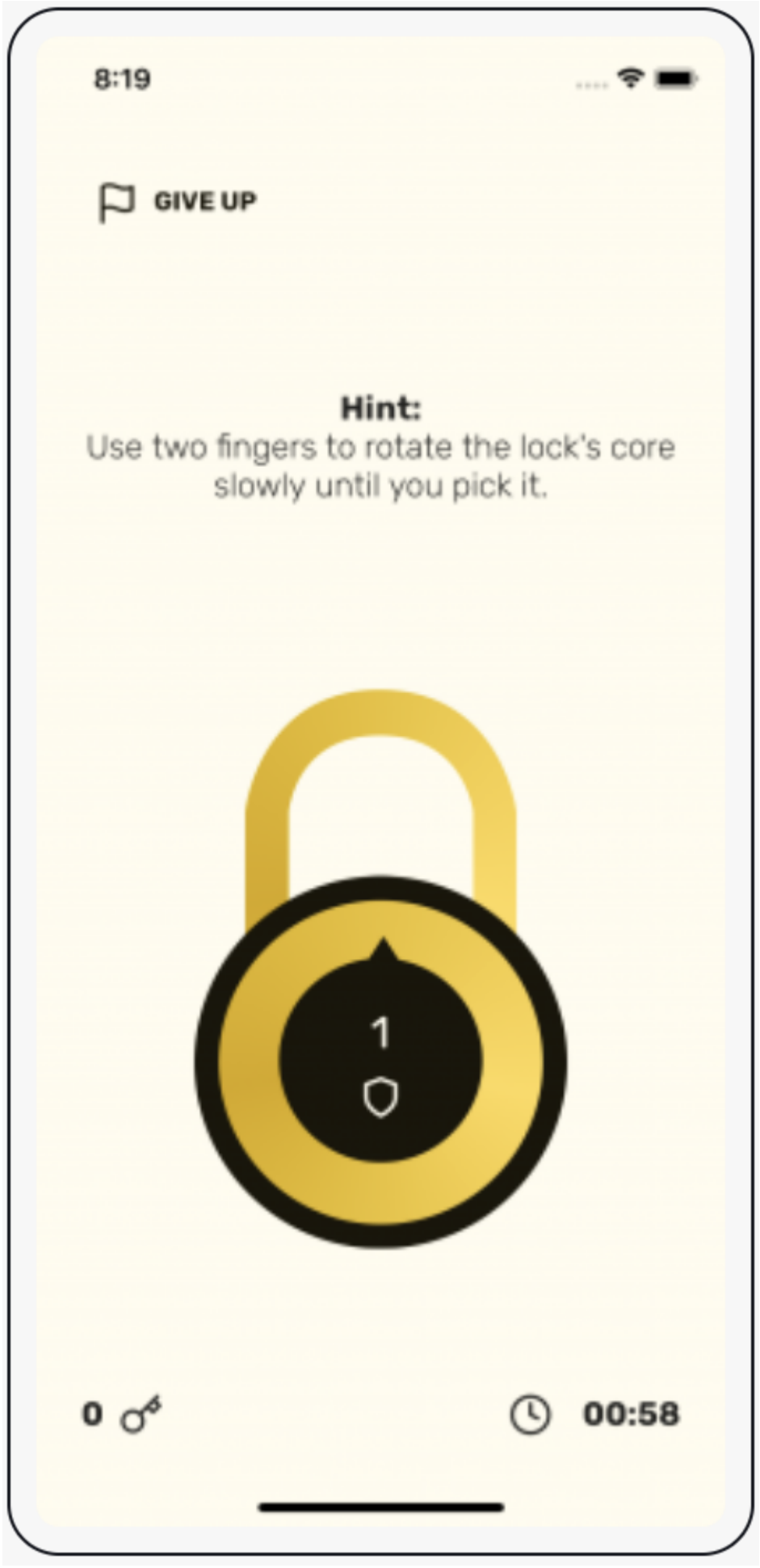
- [x] If long press the yellow rectangle, the shape of the card will change based on the value of the makeCircular property.

Incorrect

Question 7

0 / 2 pts

The following is a screenshot of a mischievous game that uses custom gestures to test the user's lock-picking skills. A rotate gesture was added to the UI, see the code below, and select the statements that are true.


8:19
GIVE UP
Hint:
Use two fingers to rotate the lock's core slowly until you pick it.
1
0
00:58

```
Widget build(BuildContext context) {
  return RawGestureDetector(
    child: child,
    gestures: <Type, GestureRecognizerFactory>{
      RotateGestureRecognizer:
      GestureRecognizerFactoryWithHandlers<RotateGestureRecognizer>(
        () => RotateGestureRecognizer()
          ..onRotationStart = onRotationStart
          ..onRotationUpdate = onRotationUpdate
          ..onRotationEnd = onRotationEnd,
          (instance) {},
        ),
}, ); }
```

```
@override
Widget build(BuildContext context) {
  final radius = MediaQuery.of(context).size.height / 4;
  // 1.
  return KeymotionGestureDetector(
    onRotationUpdate: (details) {
      setState(() {
        // 2.
        final angleDegrees = (details.rotationAngle * 180 ~/ math.pi).abs();
        currentAngle = details.rotationAngle;
        // 3.
        if (details.acceleration <= 0.0025) {
          final isCorrect = context.read<Game>().tryCombination(angleDegree
s);

          if (isCorrect) currentAngle = 0;
        }
      });
    },
    child: Stack(
      alignment: Alignment.center,
      children: [
        Transform.rotate(
          angle: ((currentAngle * 180.0) / math.pi) / 10,
          child: ConstrainedBox(
            constraints: BoxConstraints.tightForFinite(
              height: radius,
            ),
            child: Image.asset('assets/images/lock_core.png'),
          ),
        ),
        AnimatedOpacity(
          opacity: widget.combinationsLeft == 0 ? 0.4 : 1,
          duration: const Duration(milliseconds: 200),
          child: Column(
            children: [
              Text(
                widget.combinationsLeft.toString(),
                style: const TextStyle(
                  fontSize: 24,
                  color: KeyMotionColors.tint1,
                ),
              ),
              const SizedBox(height: 8),
              const Icon(
                FeatherIcons.shield,
                color: KeyMotionColors.tint1,
              ),
            ],
          ),
        ),
      ],
```

```
    ),
  );
}
```

- [ ] You should use RawGestureDetector if you're looking to use build-in gestures.

- [ ] The code uses KeymotionGestureDetector to wrap RawGestureDetector to make recognizing the rotate gesture easier to add and maintain in different parts of the app.

- [x] The code use RotateGestureRecognizer as a key in gestures, this helps GestureDetector redirect interactions to the recognizer.

- [x] The code forces the user to rotate slowly to verify the combination, by checking that the acceleration doesn't surpass an arbitrary tolerance value.

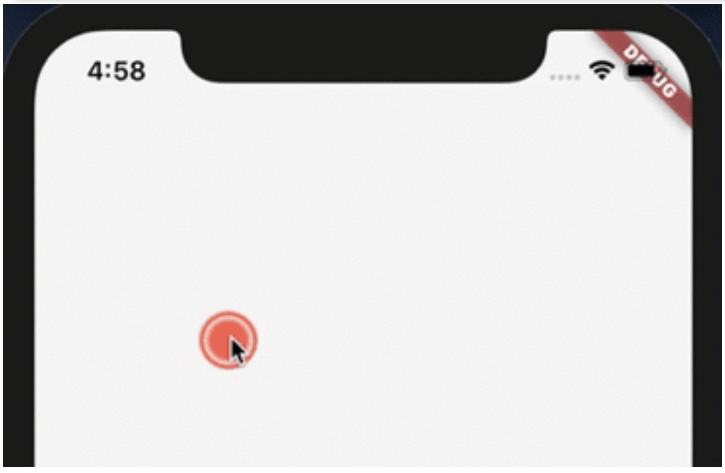
Partial

Question 8

1.33 / 2 pts

The UI (on the left is the starter UI), now you want to implement the gestures (shown on the right), The following is the code, select the statements that are true.


4:30
DEBUG
Start DX point:
Start DY point:


4:58
DEBUG


```
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
    return MaterialApp(
    title: 'Flutter Demo',
    theme: ThemeData(
      primarySwatch: Colors.grey,
    ),
    home: GestureDetectorPage(),
...

class GestureDetectorPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        color: Colors.grey[100],
        width: double.infinity,
        height: double.infinity,
        child: MainContent(),
      ),
    );
  }
}

class MainContent extends StatefulWidget {
  @override
  _MainContentState createState() => _MainContentState();
}
```

```
class _MainContentState extends State<MainContent> {
  String dragDirection = '';
  String startDXPoint = '';
  String startDYPoint = '';

  @override
  Widget build(BuildContext context) {
    return Container(
      child: Center(
          child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Text(
            this.dragDirection,
            style: TextStyle(
              fontSize: 55,
              fontWeight: FontWeight.w700,
            ),
          ),
          Text(
            'Start DX point: ${this.startDXPoint}',
            style: TextStyle(
              fontSize: 30,
              fontWeight: FontWeight.w500,
            ),
          ),
          Text(
            'Start DY point: ${this.startDYPoint}',
            style: TextStyle(
              fontSize: 30,
              fontWeight: FontWeight.w500,
            ),
          ),
        ],
      )),
    );
  }
}
```

```
/// Track starting point of a horizontal gesture
  void _onHorizontalDragStartHandler(DragStartDetails details) {
    setState(() {
      this.dragDirection = "HORIZONTAL";
      this.startDXPoint = '${details.globalPosition.dx.floorToDouble()}';
      this.startDYPoint = '${details.globalPosition.dy.floorToDouble()}';
    });
  }

  /// Track starting point of a vertical gesture
  void _onVerticalDragStartHandler(DragStartDetails details) {
    setState(() {
      this.dragDirection = "VERTICAL";
      this.startDXPoint = '${details.globalPosition.dx.floorToDouble()}';
      this.startDYPoint = '${details.globalPosition.dy.floorToDouble()}';
    });
  }
```

- [x] For getting current DX and DY position of the gesture (shown in the right screenshot) you can use either onHorizontalDragUpdate or onVerticalDragUpdate , result will be the same.

- [x] It should wrap the widget using GestureDetector.

- [x] For getting information about where the gesture started we will need to add an implementation of two functions: onHorizontalDragStart and onVerticalDragStart.

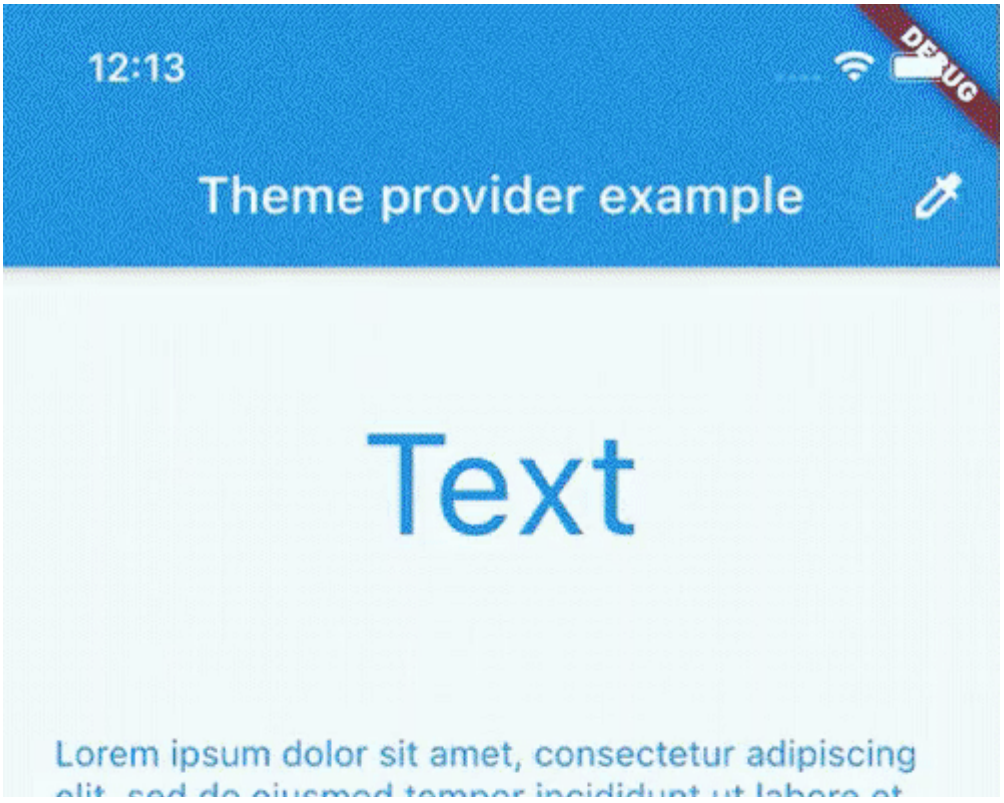
- [x] The code can also handle pinch gestures.

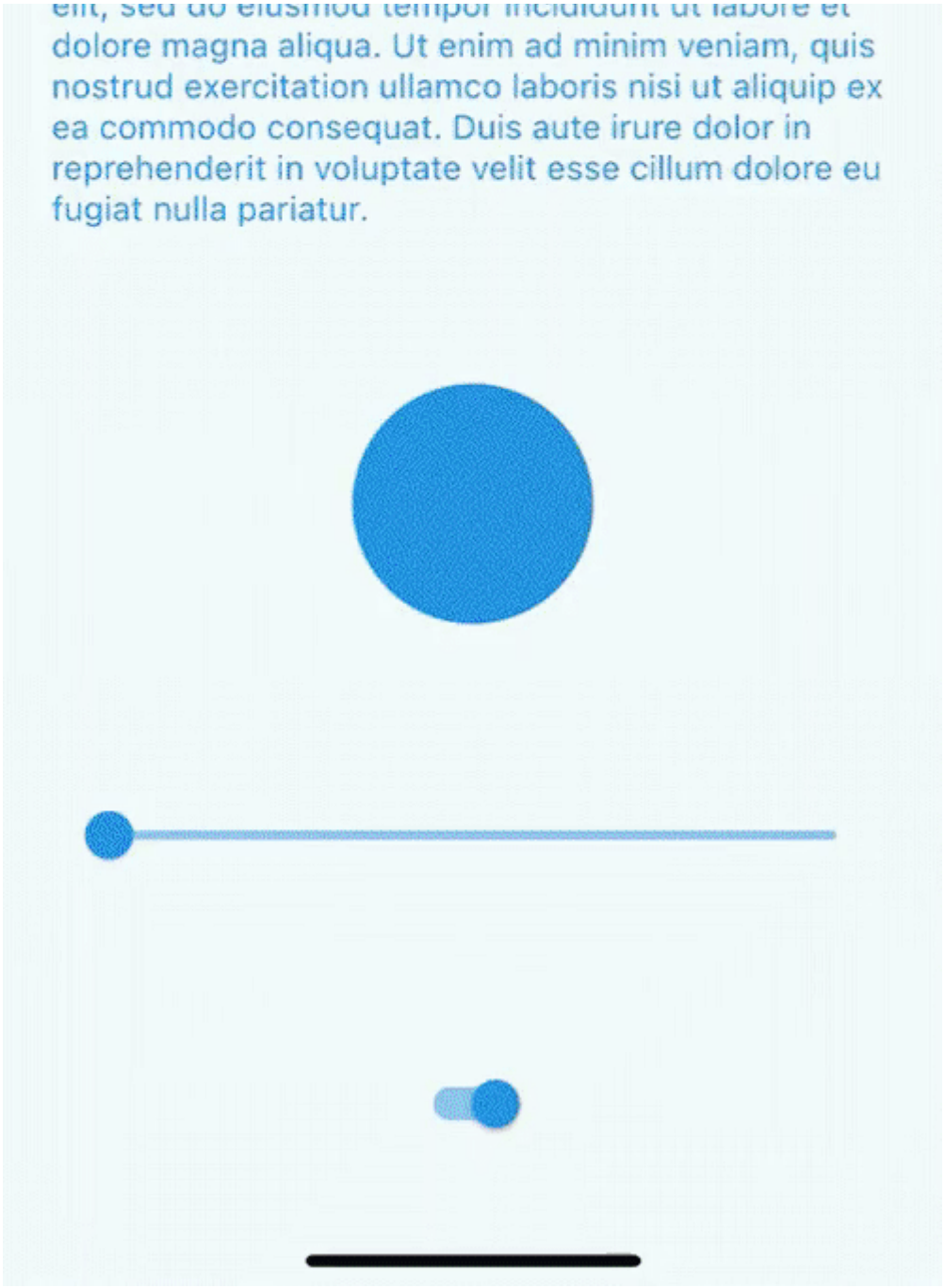
Partial

Question 9

0.67 / 1 pts

The code implemented below demonstrates how to change the main color of an app. Select all statements that are true






```
import 'package:flutter/material.dart';
import 'package:flutter_colorpicker/flutter_colorpicker.dart';
import 'package:provider/provider.dart';
import 'package:theme_provider/theme_provider.dart';
class MainScaffold extends StatelessWidget {
  const MainScaffold({Key? key}) : super(key: key);  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: _ThemedAppBar(
        title: Text('Theme provider example'),
        actions: [
          IconButton(
            icon: Icon(Icons.colorize),
            onPressed: () => _showColorPicker(context),
          ),
        ],
```

```
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.spaceAround,
          children: [
            Consumer<ThemeProvider>(
              builder: (context, themeProvider, child) => Text(
                'Text',
                style: Theme.of(context).textTheme.headline2?.copyWith(
                      color: themeProvider.mainColor,
                    ),
              ),
            ),
            Consumer<ThemeProvider>(
              builder: (context, themeProvider, child) => Text(
                'Lorem ipsum dolor sit amet, consectetur adipiscing elit, se
d do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
 minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex
 ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate v
elit esse cillum dolore eu fugiat nulla pariatur.',
                style: Theme.of(context).textTheme.bodyText2?.copyWith(
                      color: themeProvider.mainColor,
                    ),
              ),
            ),
            Consumer<ThemeProvider>(
              builder: (context, themeProvider, child) => Container(
                height: 100,
                width: 100,
                decoration: BoxDecoration(
                  color: themeProvider.mainColor,
                  borderRadius: BorderRadius.all(
                    Radius.circular(50),
                  ),
                ),
              ),
            ),
            Consumer<ThemeProvider>(
              builder: (context, themeProvider, child) => Slider(
                activeColor: themeProvider.mainColor,
                inactiveColor: themeProvider.mainColor.withOpacity(0.5),
                value: 0,
                onChanged: (newValue) {},
              ),
            ),
            Consumer<ThemeProvider>(
              builder: (context, themeProvider, child) => Switch(
                activeColor: themeProvider.mainColor,
                value: true,
                onChanged: (newValue) {},
              ),
            ),
          ],
        ),
      ),
    );
  }
  void _showColorPicker(BuildContext context) {
    ThemeProvider themeProvider =
        Provider.of<ThemeProvider>(context, listen: false);
    showDialog(
      context: context,
      builder: (context) => AlertDialog(
        titlePadding: const EdgeInsets.all(0.0),
        contentPadding: const EdgeInsets.all(0.0),
        content: Wrap(
          children: [
```

```
            ColorPicker(
              pickerColor: themeProvider.mainColor,
              onColorChanged: (color) => themeProvider.changeThemeColor(colo
r),
            ),
          ],
        ),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.of(context).pop();
            },
            child: Text('Close'),
          )
        ],
      ),
    );
  }
}
class _ThemedAppBar extends StatelessWidget with PreferredSizeWidget {
  final Widget? title;
  final List<Widget>? actions;  final Size preferredSize;  _ThemedAppBar({
    Key? key,
    this.title,
    this.actions,
  })  : preferredSize = Size.fromHeight(kToolbarHeight),
        super(key: key);  @override
  Widget build(BuildContext context) {
    return Consumer<ThemeProvider>(
      builder: (context, themeProvider, child) => AppBar(
        title: title,
        actions: actions,
        backgroundColor: themeProvider.mainColor,
      ),
    );
  }
}
```

- [ ] The themeProvider extends ChangeNotifier, so you can listen to an instance of a ChangeNotifier and being notified when it changes.

- [ ] We need to listen to changes to our ThemeProvider class so we can access the provider without a consumer like this.

- [ ] One of the main reasons to prefer Provider over Statefulwidgets is that, using Provider, you will rebuild only the widgets that needs that value (the Consumers) while the other will not be rebuilt.

When the color changes (onColorChange), we're calling the changeThemeColor of our provider that will change the color and trigger a rebuild with notifyListeners.

Question 10

1 / 1 pts

Pinch gestures are often used to zoom back out after zooming in.

- (●) True
- ( ) False

Partial

Question 11

0.67 / 1 pts

What's true about animation?

- [x] We are sensitive and prone to be distracted by meaningful motions
- [x] Animations should be used with a light touch — primarily as a tool for providing users with easily noticeable, smooth feedback.
- [x] Motion is most often appropriate as a form of subtle microinteractions to induce delight or entertain users.

- [x] Animation in UX must be unobtrusive, brief, and subtle. Use it for feedback, state-change and navigation metaphors, and to enhance signifiers.

Partial

Question 12

0.67 / 1 pts

Motion can be used for_____(Select all that apply).

- [x] Feedback
- [x] Spatial metaphors
- [x] Communication
- [x] Navigation

Question 13

1 / 1 pts

Select all the animation protocols for UX designers.

- [x] Less is more concerning animation and transitions in particular.
- [x] Timing is arguably one of the most important considerations when designing transitions.
- [x] Animations should always serve a purpose.

- [x] The motion language that you choose to use should resemble the nature of your product.

Incorrect

Question 14

0 / 1 pts

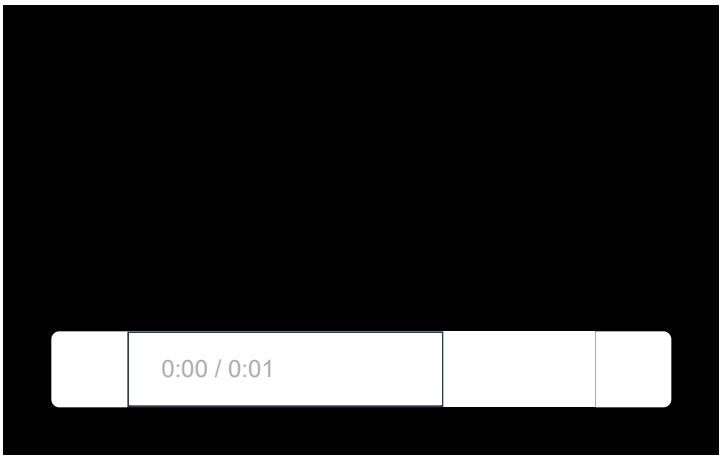
The video above is a good demonstration of _____

- (•) Motion to Communicate State Change
- ( ) Motion as a Signifier
- ( ) Attention Grabbing and Attention Hijacking
- ( ) Motion for Feedback

Partial

Question 15

0.67 / 1 pts



Select the correct answer for the above animation.

- By morphing a pencil icon into a disk after it was clicked on signals the transition from Edit to Save mode more clearly than swapping one icon out for the other instantly
- The motion was used to communicate state change.
- The motion was used for attention grabbing.
- The animation can help users by making the mode change noticeable.

Incorrect

Question 16

0 / 1 pts

What's true about the following code:

```
AnimatedContainer(
  width: 200,
  height: 200,
  color: Colors.red,
  duration: Duration(milliseconds: 250),
)
```

```
Container(
  width: 200,
  height: 200,
  color: Colors.red,
)
```

```
class AnimatedContainerPage extends StatefulWidget {
  @override
  _AnimatedContainerPageState createState() => _AnimatedContainerPageState
();
}

class _AnimatedContainerPageState extends State<AnimatedContainerPage> {
  double _width = 200;
  double _height = 200;
  Color _color = Colors.red;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: AnimatedContainer(
          width: _width,
          height: _height,
          color: _color,
          duration: Duration(milliseconds: 250),
        ),
      ),
      floatingActionButton: FloatingActionButton(
        child: Icon(Icons.play_arrow),
        onPressed: _update,
      ),
```

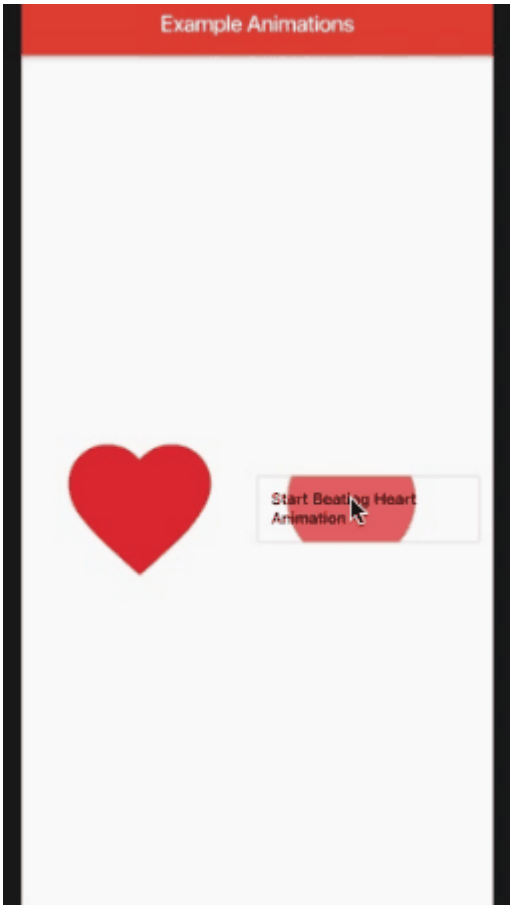
```
    );
  }
  void _update() {
    setState(() {
      _width = 300;
      _height = 300;
      _color = Colors.green;
    });
  }
}
```

- [x] The animation only works when we click the button at the first time, if we press the button again, nothing happens.
- [x] The code above uses explicitly animated widgets.
- [ ] The code above uses implicitly animated widgets.
- [ ] With the code above, we can press the button and the container animates to the new values over the given duration.

Question 17

4 / 4 pts

The code below tries to implement the following animation. Fill in the blanks in the code to make the code correct.




```
class HomeScreen extends StatefulWidget {
  @override
  _HomeScreenState createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> with TickerProviderStateMix
in {

Animation _arrowAnimation
, _heartAnimation
;
AnimationController _arrowAnimationCc
, _heartAnimationCo
;


  @override
  void initState() {
    super.initState();
    _arrowAnimationController =
        AnimationController(vsync: this, duration: Duration(milliseconds: 30
0));
    _arrowAnimation =
        Tween(begin: 0.0, end: pi).animate(_arrowAnimationController);
```

```
        _heartAnimationController = AnimationController(
        vsync: this, duration: Duration(milliseconds: 1200));
    _heartAnimation = Tween(begin: 150.0, end: 170.0).animate(CurvedAnimatio
n(
        curve: Curves.bounceOut, parent: _heartAnimationController));

    _heartAnimationController.addStatusListener((AnimationStatus status) {
      if (status == AnimationStatus.completed) {
        _heartAnimationController.repeat();
      }
    });

  }

@override
Widget build(BuildContext context) {

}
}
```

```
@override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Example Animations'),
      ),
      body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          firstChild(),
          SizedBox(
            height: 50.0,
          ),
          secondChild(),
        ],
      ),
    );
}

  Widget secondChild() {
    return Row(
      mainAxisAlignment: MainAxisAlignment.spaceAround,
      children: <Widget>[
        Expanded(
          child: AnimatedBuilder(
            animation: _heartAnimationController,
            builder: (context, child) {
              return Center(
                child: Container(
                  child: Center(
                    child: Icon(
                      Icons.favorite,
                      color: Colors.red,
                      size: _heartAnimation.value,
                    ),
                  ),
                ),
              );
            },
          ),
        ),
        Expanded(
```

```
          child: Padding(
            padding: const EdgeInsets.only(right: 12.0),
            child: OutlineButton(
              padding: const EdgeInsets.all(12.0),
              color: Colors.white,
              textColor: Colors.black,
              child: Text('Start Beating Heart Animation'),
              onPressed: () {
                _heartAnimationController.forward();
              },
              splashColor: Colors.red,
            ),
          ),
        )
      ],
    );
  }
```

Answer 1:

_arrowAnimation

Answer 2:

_heartAnimation

Answer 3:

_arrowAnimationController

Answer 4:

_heartAnimationController

Partial

Question 18

0.33 / 1 pts

What's true about the following code.

```
class FrogColor extends InheritedWidget {

  const FrogColor({

    super.key,

    required this.color,

    required super.child,

  });
```

```
  final Color color;



  static FrogColor of(BuildContext context) {
    final FrogColor? result = context.dependOnInheritedWidgetOfExactType<Fro
gColor>();
    assert(result != null, 'No FrogColor found in context');
    return result!;
  }

  @override
  bool updateShouldNotify(FrogColor old) => color != old.color;
}
```

```
class MyOtherPage extends StatelessWidget {
  const MyOtherPage({super.key});



  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: FrogColor(
        color: Colors.green,
        child: Text(
          'Hello Frog',
          style: TextStyle(color: FrogColor.of(context).color),
        ),
      ),
    );
  }
}
```

- [x] In the FrogColor class, the of method could have returned a Color instead of the FrogColor widget.

- [x] When using the of method, the context must be a descendant of the InheritedWidget

- [x] In this example, the context used is the one from the Builder, which is a child of the FrogColor widget, so this works.

- [ ] The context used is the one from the MyOtherPage widget, which is a parent of the FrogColor widget, so this does not work.

Partial

Question 19

0.33 / 1 pts

Select all the statement(s) that are true about the Provider.

- [x] By using Provider instead of manually writing InheritedWidget, you increased scalability for classes with a listening mechanism that grows exponentially in complexity.

- [x] This operation is O(n).

- [x] When using the create/update callback of a provider, this callback is called lazily by default.

Providers allow you to not only expose a value, but also create, listen, and dispose of it.

Partial

Question 20

0.67 / 1 pts

Select all the statement(s) that are true.

- [x] One-time binding is useful when only a snapshot of the data is needed, and the data is static.
- [x] Two-way binding is where changes to the data provider automatically updates the data consumer.
- [x] Data binding is a principle component of the MVVM pattern.
- [x] Data binding is the process that couples two or more data-containing elements together and synchronizes them.

Partial

Question 21

0.5 / 1 pts

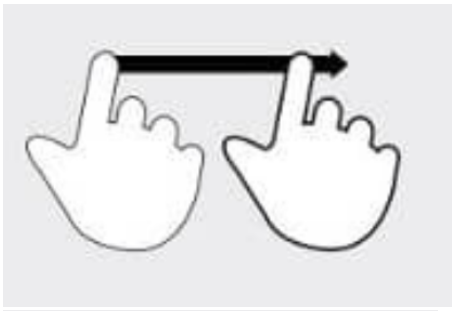
Select all the statement(s) that are true for Firebase.

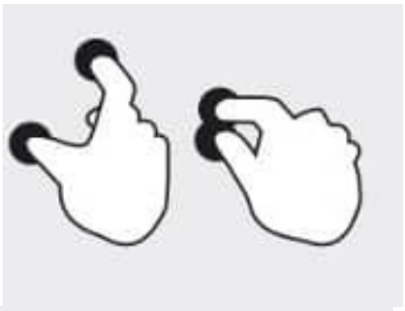
- [ ] Send users to the right place outside your app
- [x] Firebase drive organic search traffic to your app.
- [x] Firebase reduce friction with robust authentication.
- [x] Firebase store and sync app data in realtime.

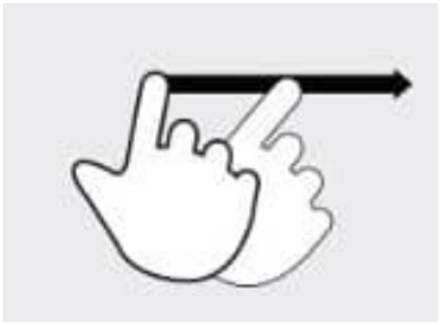
Partial

Question 22

1 / 2 pts







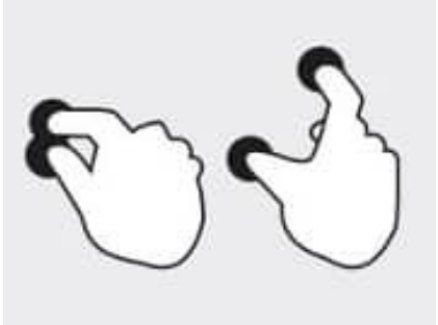


Figure 1 Figure 2
Figure 3 Figure 4

Figure 1 Drag

Figure 3 Pinch

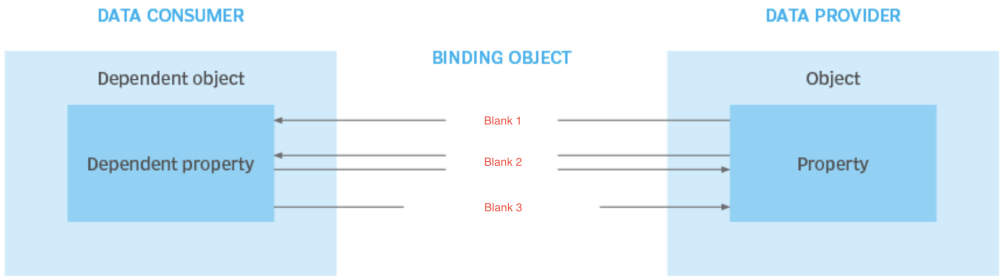
Figure 2 Flick

Figure 4 Spread

Question 23

1 / 1 pts

Matching the correct answers for the following blanks.



Blank 1 One way

Blank 2 Two way

Blank 3 One way to source

Quiz Score: **18.58** out of 30