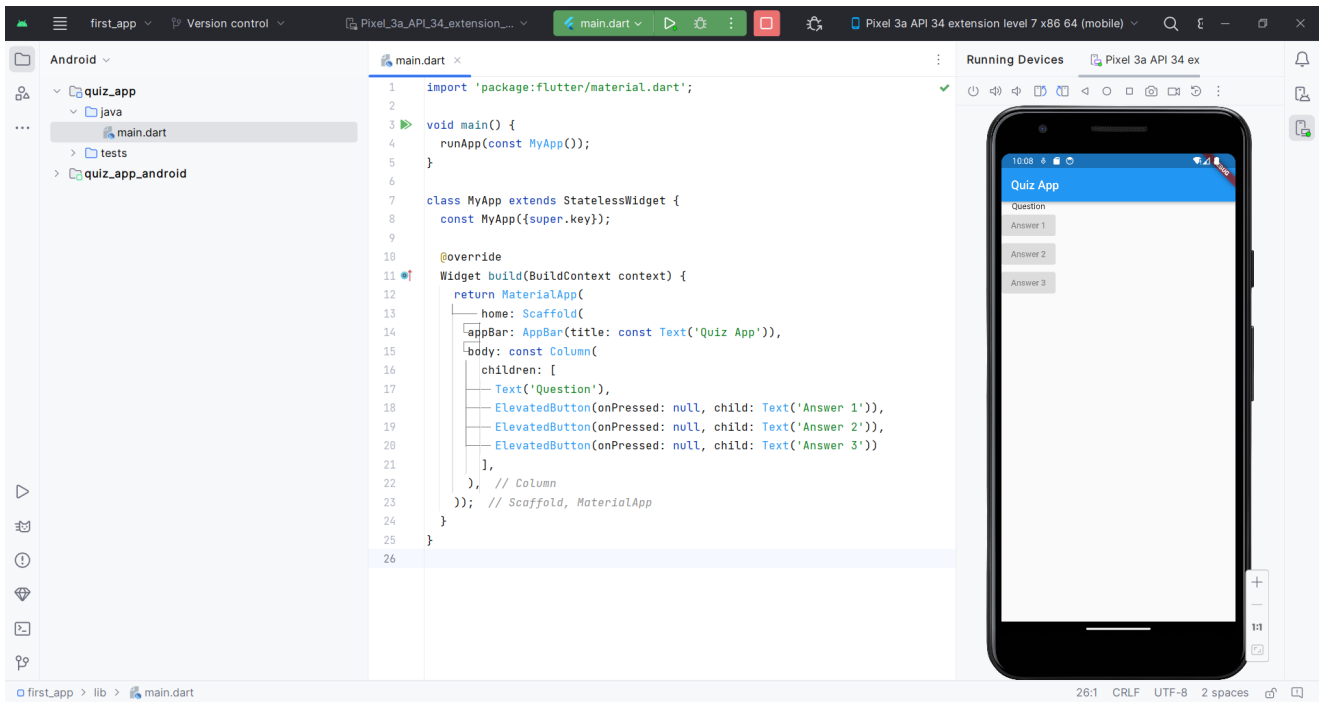


# **CSM3114 Lab 1**

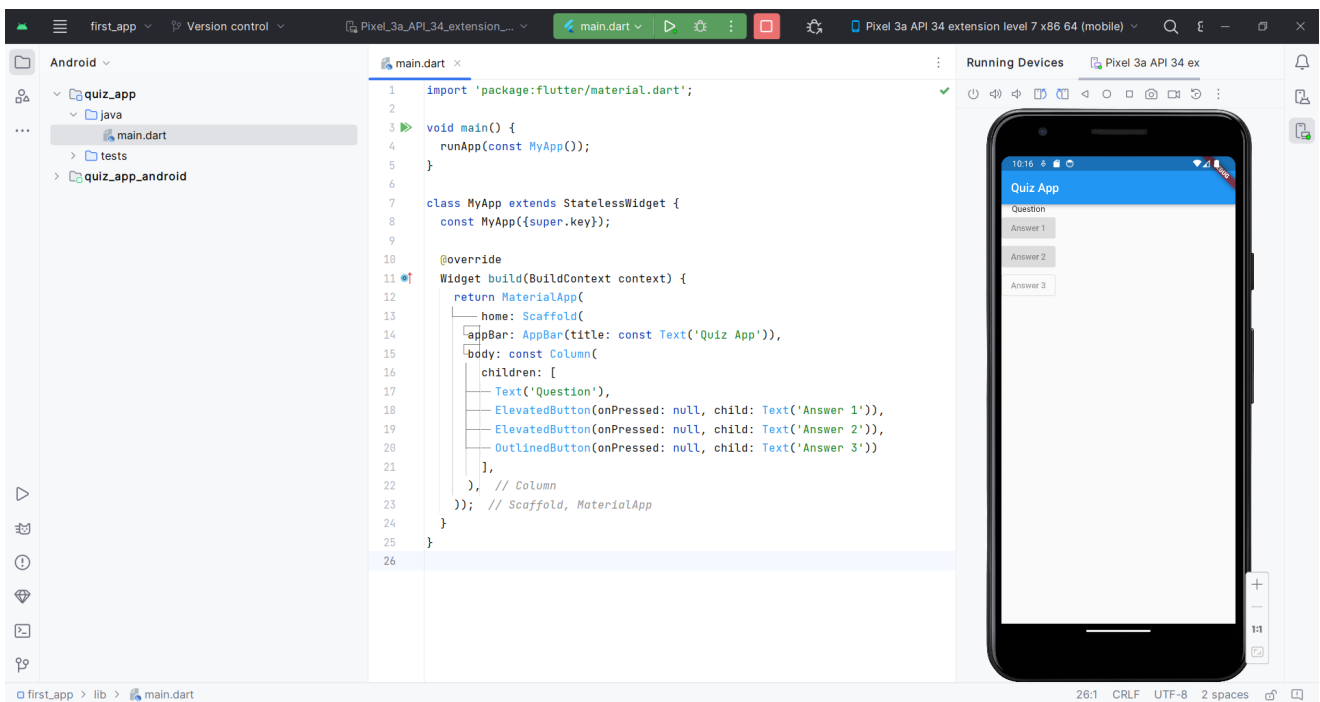
**Gary Lim S62079**

# 1 Creating a simple quiz application

## 1.2 Adding a Layout Widget for Quiz app



## 1.3 Exercise: Replace the ElevatedButton widget with OutlineButton widget



# 2 The Flutter Official Documentation

## 2.1 Finding the Flutter Official Documentation

The screenshot shows the Flutter Official Documentation for the `ElevatedButton` class. The browser address bar shows the URL `https://api.flutter.dev/flutter/material/ElevatedButton-class.html`. The page layout includes a sidebar on the left with a list of classes, a main content area with detailed information about the `ElevatedButton` class, and a sidebar on the right with a list of constructors and properties. A code sample is provided at the bottom of the main content area.

**CLASSES**

- AboutDialog
- AboutListTile
- AbsorbPointer
- Accumulator
- Action
- ActionChip
- ActionDispatcher
- ActionIconTheme
- ActionIconThemeData
- ActionListener
- Actions
- ActivateAction
- ActivateIntent
- AdaptiveTextSelectionTo...
- AlertDialog
- Align
- Alignment
- AlignmentDirectional
- AlignmentGeometry

### ElevatedButton class

A Material Design "elevated button".

Use elevated buttons to add dimension to otherwise mostly flat layouts, e.g. in long busy lists of content, or in wide spaces. Avoid using elevated buttons on already-elevated content such as dialogs or cards.

An elevated button is a label `child` displayed on a `Material` widget whose `Material.elevation` increases when the button is pressed. The label's `Text` and `Icon` widgets are displayed in `style's ButtonStyle.foregroundColor` and the button's filled background is the `ButtonStyle.backgroundColor`.

The elevated button's default style is defined by `defaultStyleOf`. The style of this elevated button can be overridden with its `style` parameter. The style of all elevated buttons in a subtree can be overridden with the `ElevatedButtonTheme`, and the style of all of the elevated buttons in an app can be overridden with the `Theme's ThemeData.elevatedButtonTheme` property.

The static `styleFrom` method is a convenient way to create a elevated button `ButtonStyle` from simple values.

If `onPressed` and `onLongPress` callbacks are null, then the button will be disabled.

This sample produces an enabled and a disabled ElevatedButton.

To create a local project with this code sample, run:

```
flutter create --sample=material.ElevatedButton.1 mysample
```

**CONSTRUCTORS**

- ElevatedButton
- icon

**PROPERTIES**

- autoFocus
- child
- clipBehavior
- enabled
- focusNode
- hashCode
- isSemanticButton
- key
- onFocusChange
- onHover
- onLongPress
- onPressed
- runtimeType
- statesController
- style

Flutter 3.13.8 • 2023-10-18 11:24 • 6c4930c4ac • stable

Show desktop

## 2.2 Exploring Text Widget

```
package:flutter/src/widgets/text.dart
Text Text(
  String data, {
  Key? key,
  TextStyle? style,
  StrutStyle? strutStyle,
  TextAlign? textAlign,
  TextDirection? textDirection,
  Locale? locale,
  bool? softWrap,
  TextOverflow? overflow,
  double? textScaleFactor,
  int? maxLines,
  String? semanticsLabel,
  TextWidthBasis? textWidthBasis,
  TextHeightBehavior? textHeightBehavior,
  Color? selectionColor,
})
```

**Containing class:** Text

Creates a text widget.

If the style argument is null, the text will use the style from the closest enclosing `DefaultTextStyle`.

The data parameter must not be null.

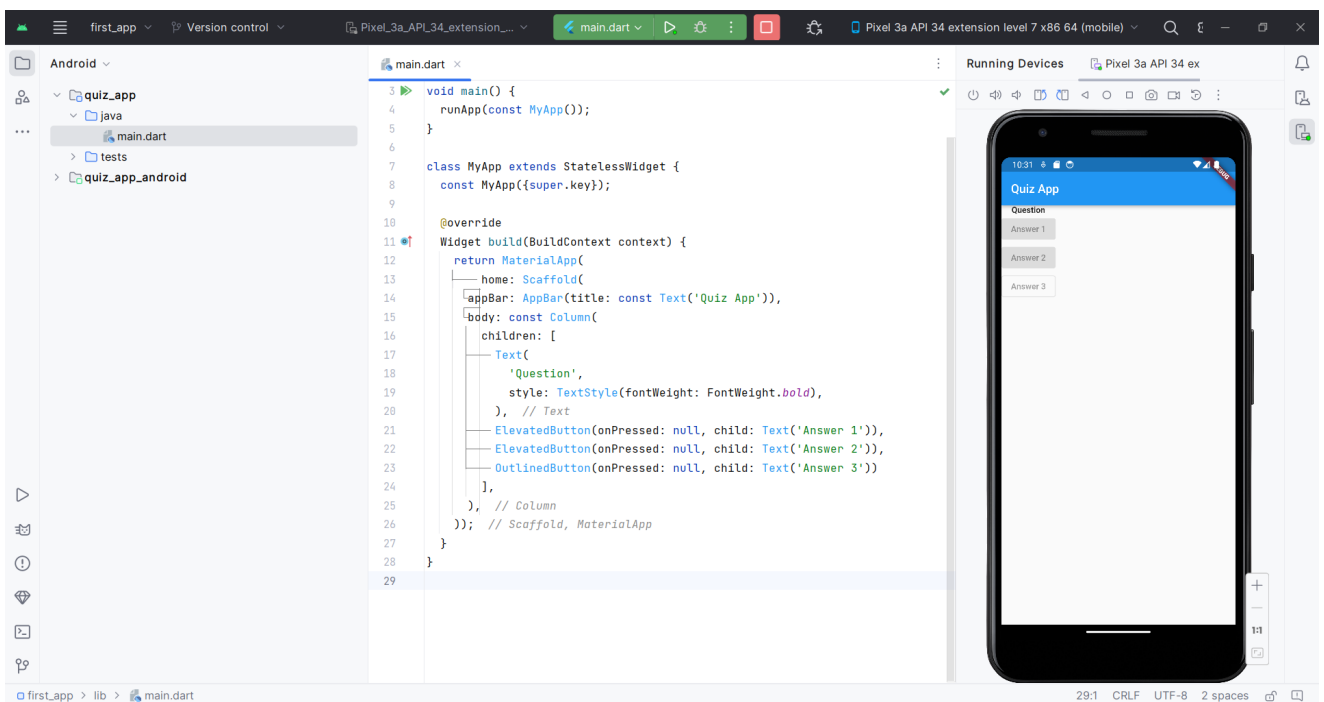
The overflow property's behavior is affected by the softWrap argument. If the softWrap is

⋮

```

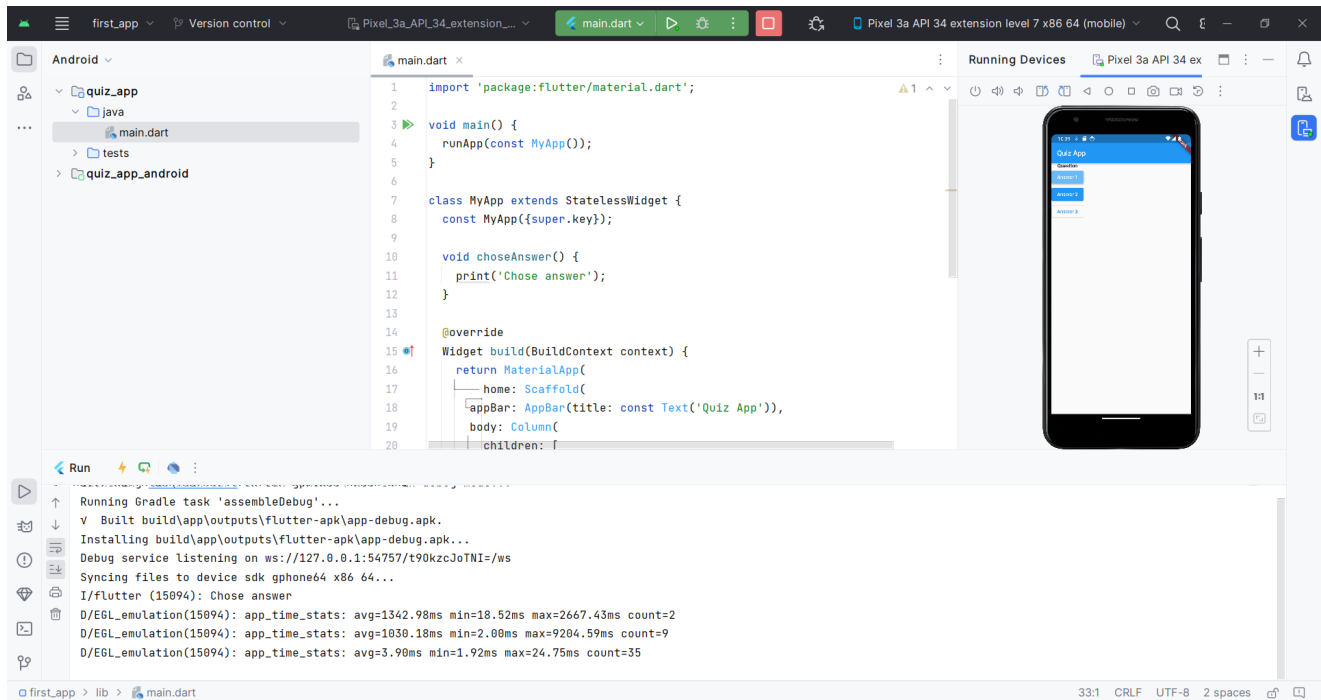
package:flutter/src/painting/text_style.dart
(const) TextStyle TextStyle({
  bool inherit = true,
  Color? color,
  Color? backgroundColor,
  double? fontSize,
  FontWeight? fontWeight,
  FontStyle? fontStyle,
  double? letterSpacing,
  double? wordSpacing,
  TextBaseline? textBaseline,
  double? height,
  TextLeadingDistribution? leadingDistribution,
  Locale? locale,
  Paint? foreground,
  Paint? background,
  List<Shadow>? shadows,
  List<FontFeature>? fontFeatures,
  List<FontVariation>? fontVariations,
  TextDecoration? decoration,
  Color? decorationColor,
  TextDecorationStyle? decorationStyle,
  double? decorationThickness,
  String? debugLabel,
  String? fontFamily,
  List<String>? fontFamilyFallback,
  String? package,
  TextOverflow? overflow,
})

```

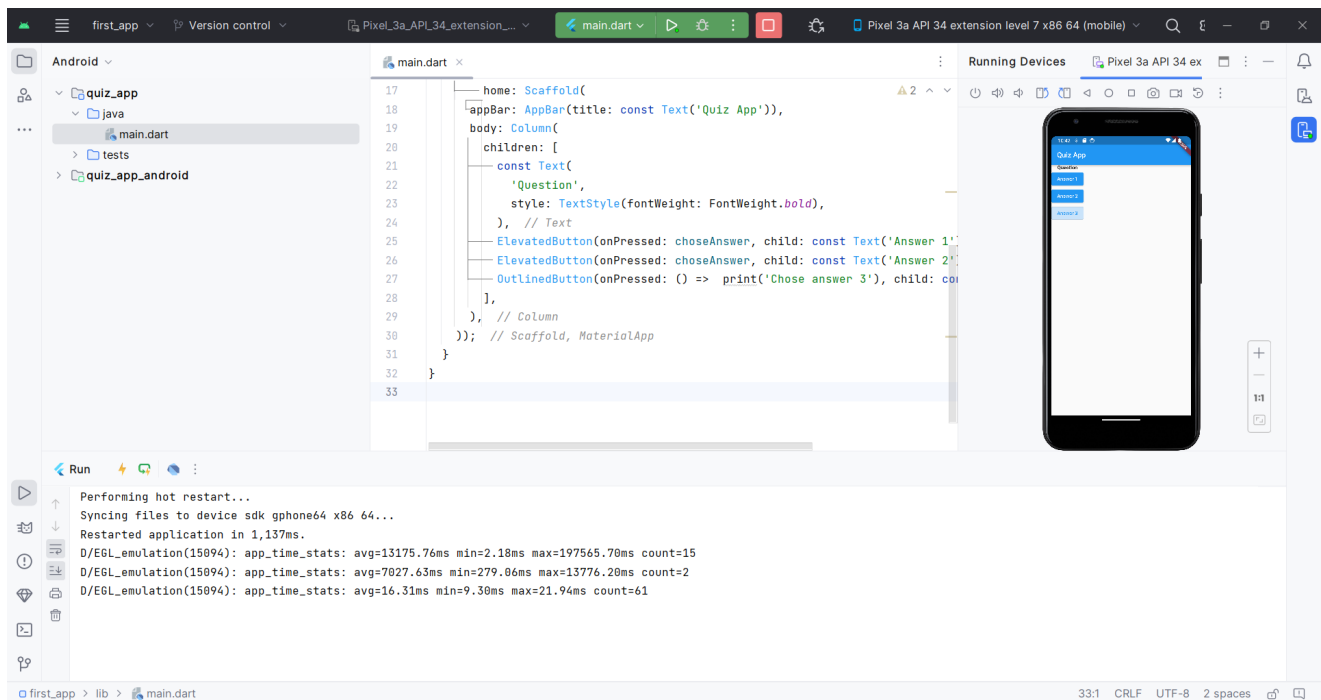


## 3 Passing Callback function

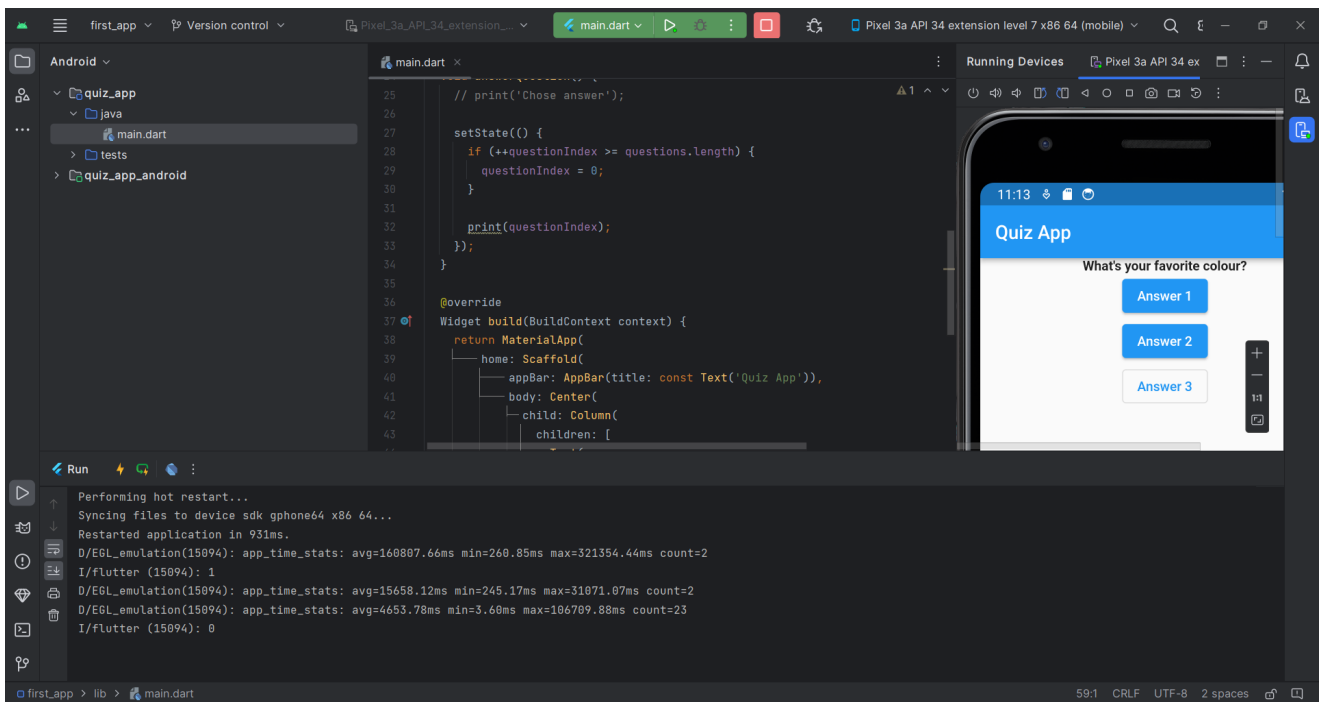
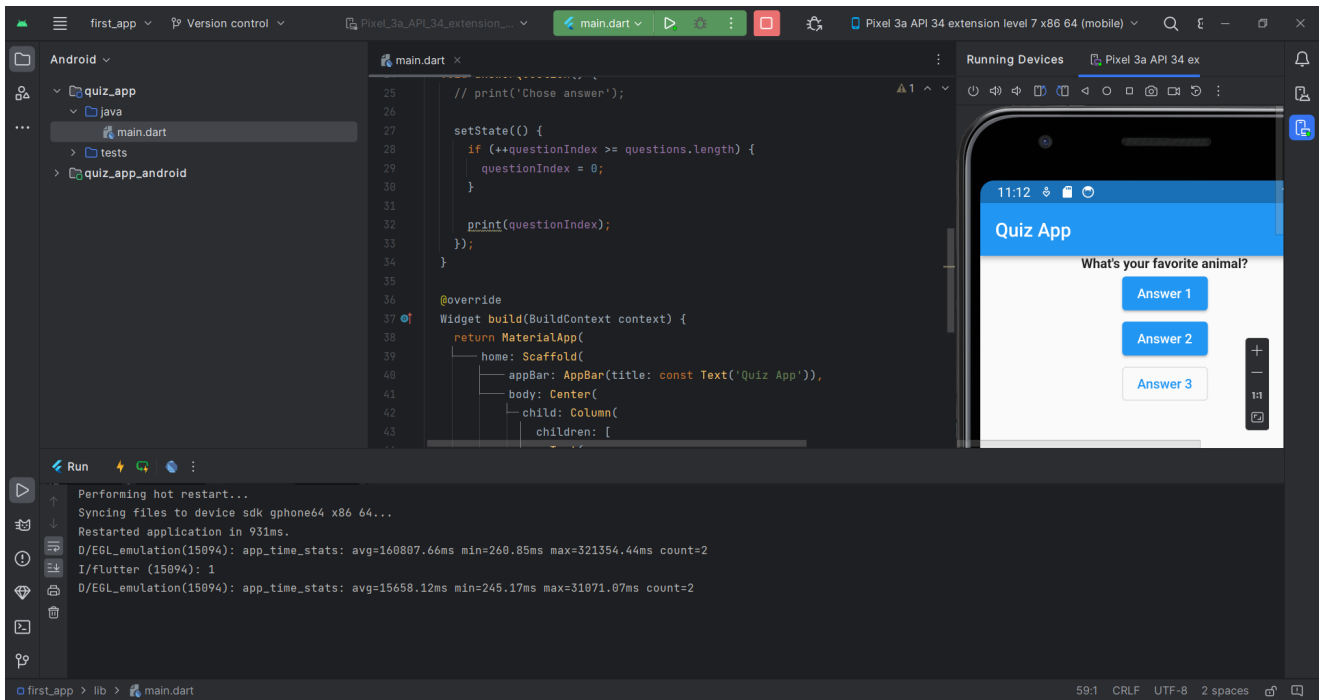
### 3.1 Creating function and attach to button



### 3.2 Using anonymous function



## 3.3 Updating Widget Data using Stateful Widget



## 3.4 Exercise