NATIONAL UNIVERSITY

NATIONAL UNIVERSITY
COLLEGE OF COMPUTING AND
INFORMATION TECHNOLOGIES

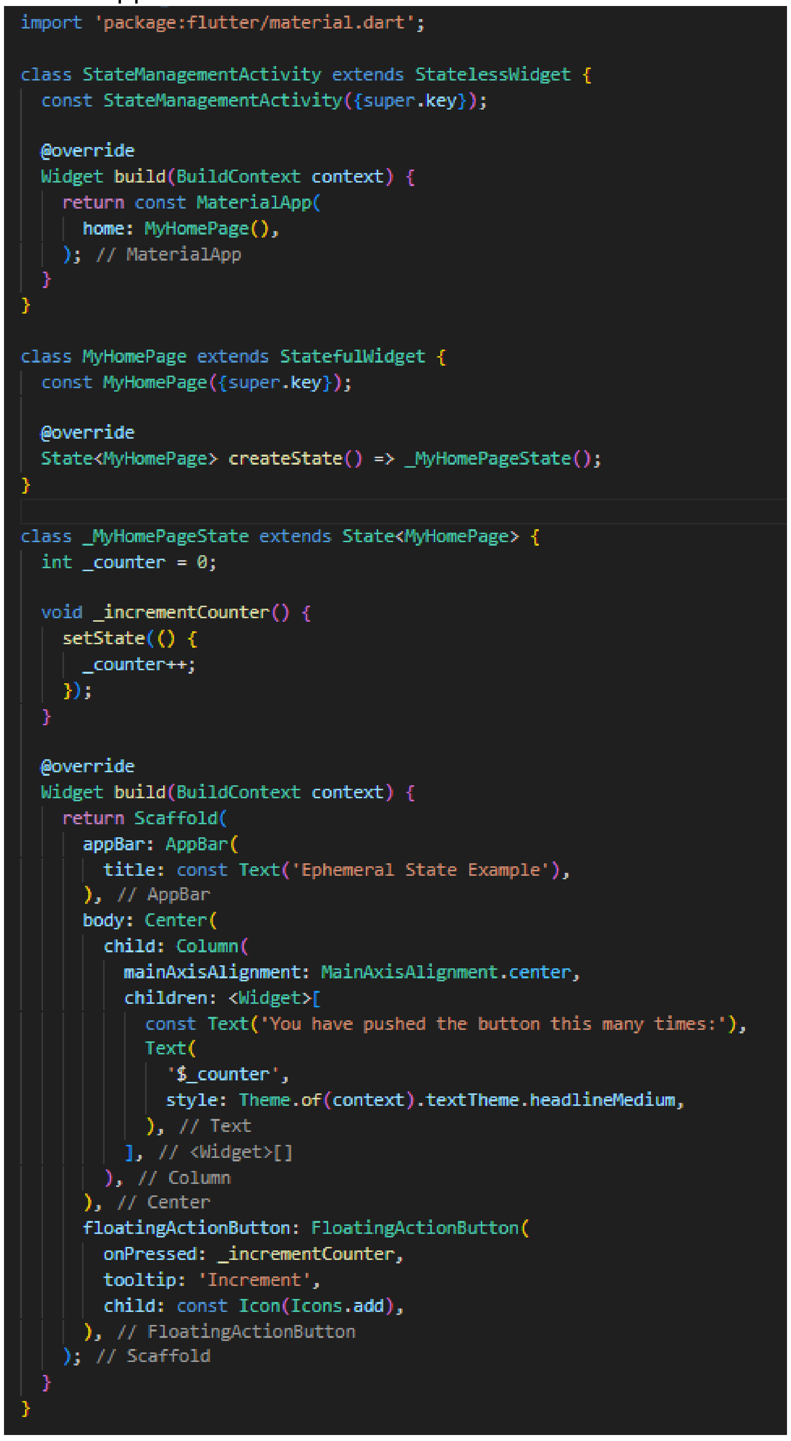
Ephemeral vs. App State

Objective:
Learn the differences between ephemeral (local) state and app state in Flutter, and how to manage them effectively in a Flutter application.
Materials Needed:

- A computer with Flutter installed.
- A code editor (like VS Code or Android Studio).
- The provider package installed in your Flutter project.

Steps:

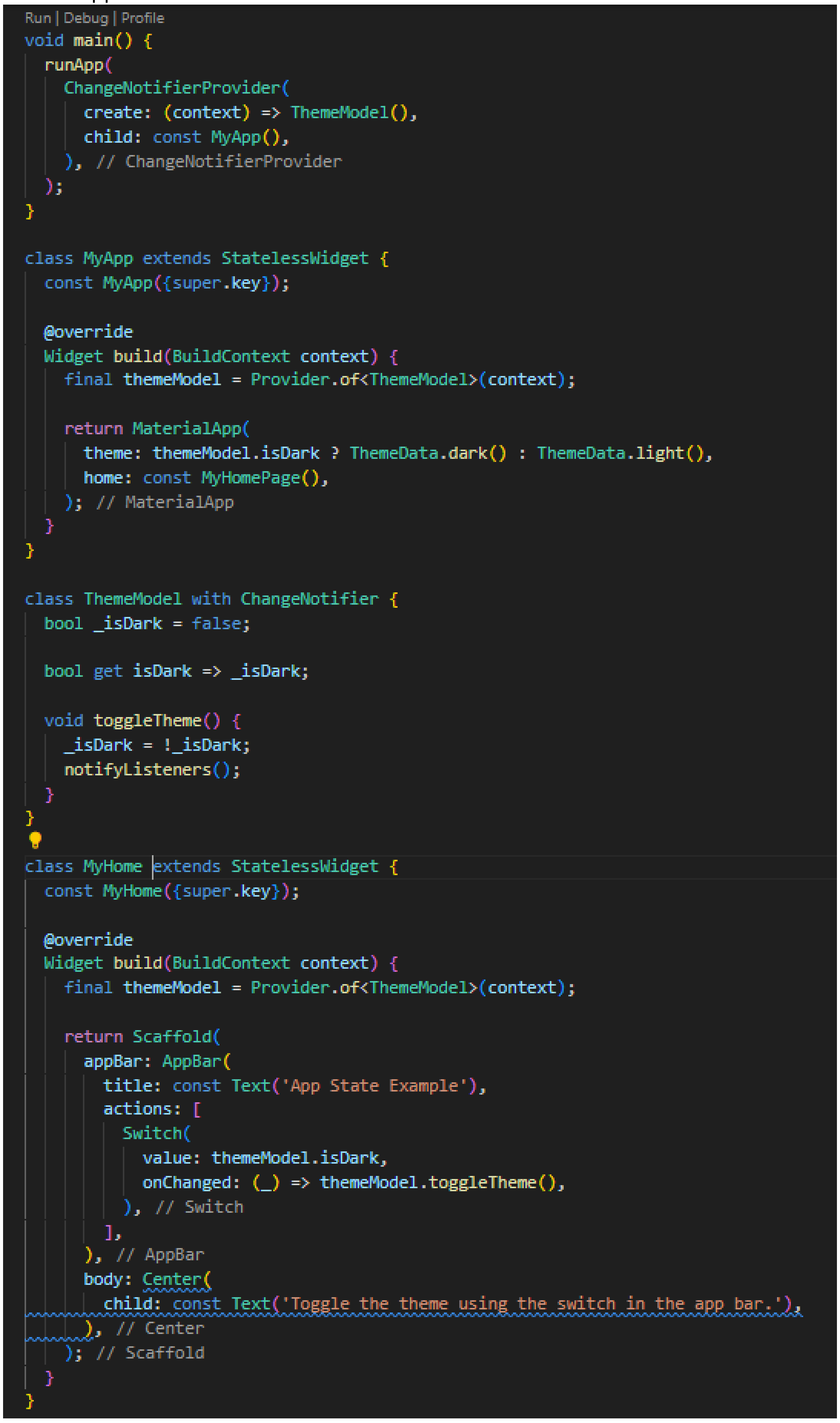
1. **Introduction:**
   - Briefly discuss what state management is in Flutter.
   - Explain the difference between ephemeral state and app state:
     - Ephemeral State: Short-lived state that only affects a specific widget or part of the UI. Example: a counter value that resets when the widget is rebuilt.
     - App State: Long-lived state that affects the entire app or large portions of the UI. Example: theme preferences (dark/light mode) that persist across different screens.
2. **Setup the Project:**
   - Create a new Flutter project using the command: flutter create ephemeral_vs_app_state.
   - Open the project in your preferred code editor.
3. **Implement Ephemeral State:**
   - In lib/main.dart, create a simple counter app that uses setState to manage the counter value.
   - Code snippet:



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

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: MyHomePage(),
    ); // MaterialApp
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key});

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Ephemeral State Example'),
      ), // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text('You have pushed the button this many times:'),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,
            ), // Text
          ], // <Widget>[]
        ), // Column
      ), // Center
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // FloatingActionButton
    ); // Scaffold
  }
}
```

   -

   - Run the app and observe how the counter behaves.
4. **Implement App State**:
   - Add the provider package to your pubspec.yaml file and run flutter pub get.
   - Modify the project to manage theme switching (dark/light mode) across the entire app using Provider.
   - Code Snippet:



```
Run | Debug | Profile
void main() {
  runApp(
    ChangeNotifierProvider(
      create: (context) => ThemeModel(),
      child: const MyApp(),
    ), // ChangeNotifierProvider
  );
}

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

  @override
  Widget build(BuildContext context) {
    final themeModel = Provider.of<ThemeModel>(context);

    return MaterialApp(
      theme: themeModel.isDark ? ThemeData.dark() : ThemeData.light(),
      home: const MyHomePage(),
    ); // MaterialApp
  }
}

class ThemeModel with ChangeNotifier {
  bool _isDark = false;

  bool get isDark => _isDark;

  void toggleTheme() {
    _isDark = !_isDark;
    notifyListeners();
  }
}

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

  @override
  Widget build(BuildContext context) {
    final themeModel = Provider.of<ThemeModel>(context);

    return Scaffold(
      appBar: AppBar(
        title: const Text('App State Example'),
        actions: [
          Switch(
            value: themeModel.isDark,
            onChanged: (_) => themeModel.toggleTheme(),
          ), // Switch
        ],
      ), // AppBar
      body: Center(
        child: const Text('Toggle the theme using the switch in the app bar.'),
      ), // Center
    ); // Scaffold
  }
}
```

5. **Hands-on Activity**:
   - Modify the app by combining both ephemeral and app state:
   - Use the counter from the first part but allow the user to toggle between light and dark themes using the switch.
6. **Discussion**:
   - Discuss the differences between using setState and Provider.

Activity Rubrics

Criteria	Excellent (4 pts)	Good (3 pts)	Fair (2 pts)	Poor (1 pt)	Points
Logic & Code Implementation	Program runs flawlessly with optimal logic flow; all features (placeholders, avatars, widgets for like/comment) are correctly implemented and error-free.	Logic is mostly correct, minor flaws present, or one minor feature missing.	Logic contains noticeable flaws or inefficiencies; two or more features missing.	Code has major logic errors, does not run, or is significantly incomplete.	
Design Accuracy & Consistency	UI matches intended design precisely with proper spacing, alignment, colors, and styles; highly consistent across all components.	Design mostly matches with only minor spacing, alignment, or style issues.	Design deviates noticeably from intended layout; inconsistent styling or incomplete visual elements.	Design is unrecognizable, lacks consistency, or ignores provided specifications.	
Widget Usage & Interactive Elements	Widgets (placeholders, avatars, like/comment buttons) are implemented effectively, enhancing functionality and interactivity.	Most widgets are used correctly with minor implementation or interaction issues.	Widgets are inconsistently used, or two or more interactive features are missing.	Widgets are not implemented, or interactivity is absent.	
Code Structure & Readability	Code is well-structured, modular, and follows best practices; easy to read, maintain, and extend.	Code is mostly organized with minor structural issues or small readability problems.	Code has structural weaknesses or poor formatting that affects readability.	Code is disorganized, hard to read, and does not follow any clear structure.	
	Other Comments/ Observations:			**Total Score**	
				Rating: (Total Score/16) * 100	