ERCİYES UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

MOBILE APPLICATION DEVELOPMENT

PROJECT ASSIGNMENT

Persistent Counter Application

Instructor: Dr. Öğr. Üyesi FEHİM KÖYLÜ

Student ID: 1030510263

Student Name: Tugay COŞKUN

TABLE OF CONTENTS

Introduction

This project involves a mobile application developed using Flutter that can persistently store the counter value for the user. The application can maintain the counter value even after the user closes and reopens the application. This feature is implemented using the SharedPreferences library.

Application Description

The "Persistent Counter" application is an enhanced version of the classic Flutter counter example. This application has the following features:

- Increment counter value
- Decrement counter value
- Reset counter value
- Store counter value in device memory

Technologies Used

The following technologies and libraries were used in the project:

- **Flutter:** The core framework of the application
- **Dart:** Programming language
- **Material Design:** User interface design language
- **SharedPreferences:** Library used to store simple data locally

Code Analysis

Application Structure

The application is created following Flutter's standard structure. The main components are:

1. **MyApp Class**: Serves as the main widget of the application and returns a MaterialApp widget.

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Persistent Counter',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const CounterPage(),
    );
  }
}
```

2. **CounterPage Class**: A StatefulWidget that represents the main screen.

```dart
class CounterPage extends StatefulWidget {
  const CounterPage({super.key});

  @override
  State<CounterPage> createState() => _CounterPageState();
}
```

3. **_CounterPageState Class**: State class that contains the counter value and related methods.

SharedPreferences Usage

The most important feature of the application is its ability to store the counter value permanently on the device. This is accomplished using the SharedPreferences library:

1. **Value Loading (_loadCounter)**: When the application is launched, the previously saved counter value is loaded.

```
Future<void> _loadCounter() async {
  final prefs = await SharedPreferences.getInstance();
  setState(() {
    _counter = prefs.getInt(_counterKey) ?? 0;
  });
}
```

2. **Value Saving (_saveCounter)**: When the counter value changes, the new value is saved to the device.

```
Future<void> _saveCounter() async {
  final prefs = await SharedPreferences.getInstance();
  await prefs.setInt(_counterKey, _counter);
}
```

Interface Design

The application interface is designed according to Material Design principles:

- An AppBar at the top
- Text displaying the counter value in the center of the screen
- Three floating buttons with different functions (increment, decrement, and reset) at the bottom

The interface code is as follows:

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      title: const Text('Persistent Counter'),
    ),
    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,
          ),
        ],
      ),
    ),
    floatingActionButton: Row(
      mainAxisAlignment: MainAxisAlignment.end,
      children: [
        FloatingActionButton(
          onPressed: _decrementCounter,
          tooltip: 'Decrement',
          heroTag: 'decrement',
          child: const Icon(Icons.remove),
        ),
        const SizedBox(width: 10),
        FloatingActionButton(
          onPressed: _resetCounter,
          tooltip: 'Reset',
          heroTag: 'reset',
          child: const Icon(Icons.refresh),
        ),
        const SizedBox(width: 10),
        FloatingActionButton(
          onPressed: _incrementCounter,
          tooltip: 'Increment',
          heroTag: 'increment',
          child: const Icon(Icons.add),
        ),
      ],
    ),
  );
}
```

Application Functions

The application supports three basic functions:

1. **Counter Increment (_incrementCounter)**: Increases the counter value by one and saves the new value.

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

2. **Counter Decrement (_decrementCounter)**: Decreases the counter value by one and saves the new value.

```
void _decrementCounter() {
  setState(() {
    _counter--;
    _saveCounter();
  });
}
```

3. **Counter Reset (_resetCounter)**: Resets the counter value to zero and saves the new value.

```
void _resetCounter() {
  setState(() {
    _counter = 0;
    _saveCounter();
  });
}
```

Conclusion

This project provides a simple but effective example to demonstrate the local data storage concept in Flutter. The use of SharedPreferences is an effective method for persistently storing small amounts of data, and this feature has been successfully implemented in this project.

The application also demonstrates Flutter's widget system and state management. A dynamic interface was created using StatefulWidget, and user interactions (button clicks) were handled correctly.