btn_to_setting.dart

소스 코드

전체 코드

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

import '../../config.dart';
import '../../router.dart';

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

  @override
  State<ToSettingButton> createState() => _ToSettingButtonState();
}

class _ToSettingButtonState extends State<ToSettingButton> {
  int _tapCount = 0;
  DateTime? _lastTapTime;
  final int _requiredTaps = 5;
  final Duration _timeLimit = const Duration(seconds: 10);
  final String _secretCode = "1111";

  void _handleTap() {
    print("_handleTap");
    final now = DateTime.now();

    if (_lastTapTime == null || now.difference(_lastTapTime!) > _timeLimit) {
      _tapCount = 1;
      print("Tap count reset. Current tap: $_tapCount");
    } else {
      _tapCount++;
      print("Tap count: $_tapCount");
    }

    _lastTapTime = now;

    if (_tapCount >= _requiredTaps) {
      _tapCount = 0; // 카운트 초기화
      _lastTapTime = null; // 시간 초기화
      print("Required taps met. Showing dialog.");
      _showSecretDialog();
    }
  }

  Future<void> _showSecretDialog() async {
    final TextEditingController controller = TextEditingController();
    final result = await showDialog<String>(
```

```
      context: context,
      barrierDismissible: false, // 다이얼로그 바깥을 탭해도 닫히지 않도록
      builder: (BuildContext context) {
        return AlertDialog(
          title: const Text('Input Password'),
          content: TextField(
            controller: controller,
            keyboardType: TextInputType.number,
            obscureText: true, // 비밀번호처럼 보이도록 (선택 사항)
            decoration: const InputDecoration(hintText: '4 Digits'),
            maxLength: 4,
          ),
          actions: <Widget>[
            TextButton(
              child: const Text('Cancel'),
              onPressed: () {
                Navigator.of(context).pop(); // 다이얼로그 닫기
              },
            ),
            TextButton(
              child: const Text('Ok'),
              onPressed: () {
                Navigator.of(context).pop(controller.text); // 입력된 값 반환하며 닫기
              },
            ),
          ],
        );
      },
    );

    if (result == _secretCode) {
      print("Secret code matched. Navigating to SecretPage.");
      if (mounted) { // 위젯이 여전히 마운트되어 있는지 확인
        context.pushNamed(RouteGroup.Setting.name);
      }
    } else if (result != null) {
      print("Secret code does not match. Entered: $result");
      if (mounted) {
        ScaffoldMessenger.of(context).showSnackBar(
          const SnackBar(content: Text('비밀번호가 일치하지 않습니다.')),
        );
      }
    }
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      behavior: HitTestBehavior.opaque,
      onTap: _handleTap,
      child: Config.instance.isDebugMode ?
    Container(
    alignment: Alignment.center,
```

```
      width: 300,
      height: 300,
      child: Center(
        child: Text("Move To Setting Page Button[DebugMode]"),
      ),
      color: Colors.red.withAlpha(40),
    ) : Container(
      alignment: Alignment.center,
      width: 300,
      height: 300,
      )
    );
  }
}
```

1. 구조 분석 (Structural Analysis)

1.1 상태 관리 구성

컴포넌트는 다음 상태를 보유합니다.

- `_tapCount` : 현재까지 입력된 탭 횟수
- `_lastTapTime` : 마지막 입력 시간
- `_requiredTaps` : Secret 기능 활성화를 위한 요구 횟수 (5회)
- `_timeLimit` : 연속 입력 가능 시간 (10초)
- `_secretCode` : 사용자 입력 검증용 4자리 코드(1111)

구조는 `ConnectPortButton`과 동일한 Secret Activation 패턴을 따릅니다.

1.2 입력 처리 흐름

핵심 로직 `_handleTap()`는 다음과 같은 조건 흐름을 갖습니다.

1. 일정 시간 경과 시 카운트 초기화
2. 연속 입력이면 카운트 증가
3. 요구 횟수 충족 시 Secret Dialog 호출

로직 분리는 적절하나, reset 동작이 반복적으로 등장하여 함수 분리가 필요합니다.

1.3 Secret Dialog 흐름

`showDialog → controller 입력 → 팝업 종료 → 코드 비교 → Routing` 흐름은 명확합니다.
그러나 다음 개선 여지가 존재합니다.

- TextEditingController는 Dialog 생성 시점에만 필요 → **Stateless 생성**되어 있어 적절함.
- `result != null` 조건 처리 시, “Cancel”의 경우까지 SnackBar 노출 가능 → 불필요한 조건 분기 필요.

1.4 UI Layer

현재 UI는 아래 형태입니다.

```
GestureDetector(
  behavior: HitTestBehavior.opaque,
  onTap: _handleTap,
  child: ...
)
```

버튼임에도 InkWell 효과가 없어 사용자가 피드백을 받기 어렵습니다.
또한 디버그/릴리즈의 분기 구조가 반복되고 있어 추상화 가능성이 있습니다.

---

2. 개선 포인트 (Issues & Improvements)

개선 요소	설명
GestureDetector → InkWell 교체	버튼 시각/터치 피드백 제공
_resetTapState() 분리	중복 로직 제거
dialog 반환값 처리 개선	Cancel 케이스 스낵바 노출 방지
디버그 모드 UI 정리	스타일 일관성 확보
provider/라우팅은 현 상태 유지	구조적 문제 없음

---

3. ✔ Step3 — 완성된 리팩토링 버전 (정식 문체)

공식 스타일로 재구성한 최종 버전입니다.

```
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import '../../config.dart';
import '../../router.dart';

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

  @override
  State<ToSettingButton> createState() => _ToSettingButtonState();
}

class _ToSettingButtonState extends State<ToSettingButton> {
  static const int _requiredTaps = 5;
  static const Duration _timeLimit = Duration(seconds: 10);
  static const String _secretCode = "1111";

  int _tapCount = 0;
  DateTime? _lastTapTime;
```

```
20
21  @override
22  Widget build(BuildContext context) {
23    final bool debug = Config.instance.isDebugMode;
24
25    return Material(
26      color: Colors.transparent,
27      child: InkWell(
28        onTap: _handleTap,
29        borderRadius: BorderRadius.circular(8),
30        child: Container(
31          alignment: Alignment.center,
32          width: 300,
33          height: 300,
34          color: debug ? Colors.red.withAlpha(40) : Colors.transparent,
35          child: debug
36              ? const Text(
37                  "Move To Setting Page Button [DebugMode]",
38                  style: TextStyle(fontSize: 16),
39                )
40              : null,
41        ),
42      ),
43    );
44  }
45
46  void _handleTap() {
47    final now = DateTime.now();
48
49    if (_lastTapTime == null || now.difference(_lastTapTime!) > _timeLimit) {
50      _tapCount = 1;
51    } else {
52      _tapCount++;
53    }
54
55    _lastTapTime = now;
56
57    if (_tapCount >= _requiredTaps) {
58      _resetTapState();
59      _showSecretDialog();
60    }
61  }
62
63  void _resetTapState() {
64    _tapCount = 0;
65    _lastTapTime = null;
66  }
67
68  Future<void> _showSecretDialog() async {
69    final controller = TextEditingController();
70
71    final result = await showDialog<String>(
72      context: context,
```

```
      barrierDismissible: false,
      builder: (context) {
        return AlertDialog(
          title: const Text('Input Password'),
          content: TextField(
            controller: controller,
            keyboardType: TextInputType.number,
            obscureText: true,
            maxLength: 4,
            decoration: const InputDecoration(hintText: '4 Digits'),
          ),
          actions: [
            TextButton(
              child: const Text('Cancel'),
              onPressed: () => Navigator.of(context).pop(),
            ),
            TextButton(
              child: const Text('Ok'),
              onPressed: () => Navigator.of(context).pop(controller.text),
            ),
          ],
        );
      },
    );

    // Cancel 또는 null = 처리하지 않음
    if (result == null || result.isEmpty) {
      return;
    }

    if (result == _secretCode) {
      if (!mounted) return;
      context.pushNamed(RouteGroup.Setting.name);
      return;
    }

    if (!mounted) return;
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('비밀번호가 일치하지 않습니다.')),
    );
  }
}
```

4. 리팩토링 결과 요약

개선 항목	상태
Material + InkWell 전환	✔
Secret 입력 처리 개선	✔

개선 항목	상태
Reset 로직 분리	✔
Cancel 시 SnackBar 제거	✔
디버그/릴리즈 UI 정리	✔
코드 구조 명료화	✔