

Report

Planity-Application

(นางสาววิชิตา ใจดี หมายเลขพานิช 660710078)

แอปพลิเคชันนี้ถูกออกแบบมาเพื่อเป็นเครื่องมือจัดการงาน (To-do List) และกำหนดการ (Event/Deadline) โดยใช้ Flutter เป็น Frontend และ Firebase เป็น Backend หลัก

1. โครงสร้าง (Architecture)

แอปพลิเคชันนี้มีการจัดโครงสร้างไฟล์ โดยแบ่งเป็นกลุ่มหลักๆ 4 กลุ่ม ทำให้ติดตามและจัดการง่ายดังนี้

Folders	จำนวนไฟล์	หน้าที่หลัก	ชื่อไฟล์	หน้าที่คร่าวๆ
Models	3	ใช้ในการ กำหนดรูปแบบ ของ ข้อมูลที่ได้รับจาก Firestore และ ใช้ในการ ส่งข้อมูล กลับไปยัง Firestore	event_model.dart	กำหนดโครงสร้างข้อมูลสำหรับ Event (กำหนดการ/ Deadline) พร้อมฟังก์ชันแปลงข้อมูลระหว่าง Dart Object กับ Firestore Map
			model.dart	กำหนดโครงสร้างข้อมูลสำหรับ Todo (รายการสิ่งที่ต้องทำ) และ TodoGroup พร้อมฟังก์ชันแปลงข้อมูลระหว่าง Dart Object กับ Firestore Map
			todo_model.dart	เป็นไฟล์ Exporter ที่รวมเอา event_model.dart และ todo_model.dart เข้าด้วยกัน เพื่อให้ไฟล์อื่นๆ สามารถ Import โครงสร้างข้อมูลทั้งหมดได้ด้วยคำสั่งเดียว
Services	5 (+1 Exporter)	ใช้ในการ สื่อสาร กับ Firebase/Firestore เพื่อทำคำสั่ง CRUD และจัดการตระหนักระยะ ทาง	auth_service.dart	จัดการ Authentication ทั้งหมด เช่น Sign In, Register, Sign Out, Delete Account, และการจัดการข้อมูล ผู้ใช้ที่มาจากการ Firebase Auth
			event_service.dart	จัดการ CRUD (Create, Read, Update, Delete) สำหรับ ข้อมูล Events ใน Firestore รวมถึงการดึงข้อมูลแบบ Real- time (Stream)
			todo_service.dart	จัดการ CRUD สำหรับข้อมูล Todos และ Todo Groups ใน Firestore รวมถึงการจัดการลำดับ (Reorder) และสถานะเสร็จ สิ้น (Toggle Complete)
			notification_service.dart	จัดการ Logic ในการดึงข้อมูล Events ที่กำลังจะถึงกำหนด (วันนี้/พรุ่งนี้) จาก Firestore เพื่อนำมาแสดงเป็นการแจ้งเตือน และนับจำนวน Badge ใน Navigation Bar
			settings_service.dart	จัดการการตั้งค่า User Settings (เช่น เปิด/ปิด Notification) และจัดการการอัปเดตข้อมูล User Profile (เช่น ชื่อ) ใน Firestore
			services.dart	เป็นไฟล์ Exporter ที่รวม *service.dart ทุกไฟล์เข้าด้วยกัน เพื่อให้ไฟล์อื่นๆ สามารถ Import โครงสร้างข้อมูลทั้งหมดได้ ด้วยคำสั่งเดียว

Pages	7 (+1 Exporter)	หน้าจอหลักของแอปพลิเคชัน นำ Widgets มาประกอบกัน เป็น View	begin_page.dart	หน้าจอแรกสุดที่ผู้ใช้เห็น (Landing Page) มีปุ่มให้เลือก Login หรือ Register
			login_page.dart	หน้าจอสำหรับผู้ใช้ที่ต้องการ เข้าสู่ระบบ โดยเรียกใช้ AuthService ในการตรวจสอบสิทธิ์
			register_page.dart	หน้าจอสำหรับผู้ใช้ที่ต้องการ ลงทะเบียนบัญชีใหม่ โดยเรียกใช้ AuthService เพื่อสร้างบัญชีและ User Document
			welcome_page.dart	หน้าจอที่แสดงให้เห็นหลังจากเข้าสู่ระบบครั้งแรก เพื่อให้ผู้ใช้ ก่อนเข้าสู่หน้าหลัก
			home_page.dart	หน้าจอหลัก ของแอป แสดงรายการ Todos และ Events มี พิงค์ชั้นหลักในการเพิ่ม/แก้ไข/ลบ และแสดง Quote of the Day https://corsproxy.io/?https://zenquotes.io/api/today (API)
			notification_page.dart	หน้าจอดูรายการ การแจ้งเตือน (Events ที่จะถึงกำหนดในวันนี้/พรุ่งนี้) โดยดึงข้อมูลมาจาก NotificationService
			setting_page.dart	หน้าจอสำหรับ ตั้งค่า ต่างๆ ของผู้ใช้ เช่น การเปิด/ปิด Notification, การดูข้อมูลส่วนตัว, และปุ่ม Sign Out/Delete Account
			pages.dart	เป็นไฟล์ Exporter ที่รวม *page.dart ทุกไฟล์ไว้ด้วยกัน
Widgets	12 (+1 Exporter)	ส่วนประกอบ UI ที่นำกลับมาใช้ซ้ำได้ เช่น ปุ่ม, ช่องกรอก, Card, Dialog	button_custom.dart	สร้าง ปุ่ม ที่มีการออกแบบ (สี, เก้า, รูปแบบ) เพื่อใช้ในหน้าจอต่างๆ (Login, Register, Dialogs)
			custom_text_field.dart	สร้าง ช่องกรอกข้อความ ที่มีการออกแบบ รองรับการตรวจสอบ (Validator) และมีฟังก์ชันซ่อน/แสดงรหัสผ่าน
			custom_dialog.dart	มีฟังก์ชันสำหรับแสดง Dialog ที่ว่าไป เช่น showError, showSuccess, และ showLoading
			custom_link_text.dart	สร้างข้อความที่เมื่อสัมผัสถูกคลิก (Tap) จะเปลี่ยนเป็น สีเงิน (เช่น "Don't have an account? Register")
			gradient_background.dart	สร้างพื้นหลังแบบไลส์ (Gradient) ที่ใช้ครอบหน้าจอหลัก (Pages) เพื่อให้มีดีไซน์ที่สวยงาม
			page_title.dart	สร้าง ข้อความชื่อหน้าจอ ในรูปแบบมาตรฐาน (ตัวใหญ่, Bold)
			main_navigation.dart	สร้าง Bottom Navigation Bar หลักของแอป ซึ่งใช้ในการเปลี่ยนไปมาระหว่าง Home, Notification, และ Settings พร้อมแสดง Badge นับการแจ้งเตือน
			todo_card.dart	สร้าง Card สำหรับแสดงข้อมูล Todo แต่ละรายการในหน้า Home รองรับการ Check/Uncheck และ Menu Edit/Delete
			event_card.dart	สร้าง Card สำหรับแสดงข้อมูล Event แต่ละรายการในหน้า Home พร้อมแสดง แคบสี ตามประเภทของ Event และ Menu Edit/Delete
			todo_dialog.dart	สร้าง Dialog สำหรับ เพิ่ม/แก้ไข รายการ Todo รวมถึงการเลือก Emoji และ Group
			event_dialog.dart	สร้าง Dialog สำหรับ เพิ่ม/แก้ไข รายการ Event รวมถึงการเลือก Deadline และ Border Color
			group_dialog.dart	สร้าง Dialog สำหรับ เพิ่ม Group ของ Todo โดยเฉพาะ
			widgets.dart	เป็นไฟล์ Exporter ที่รวม *widget.dart ทุกไฟล์ไว้ด้วยกัน

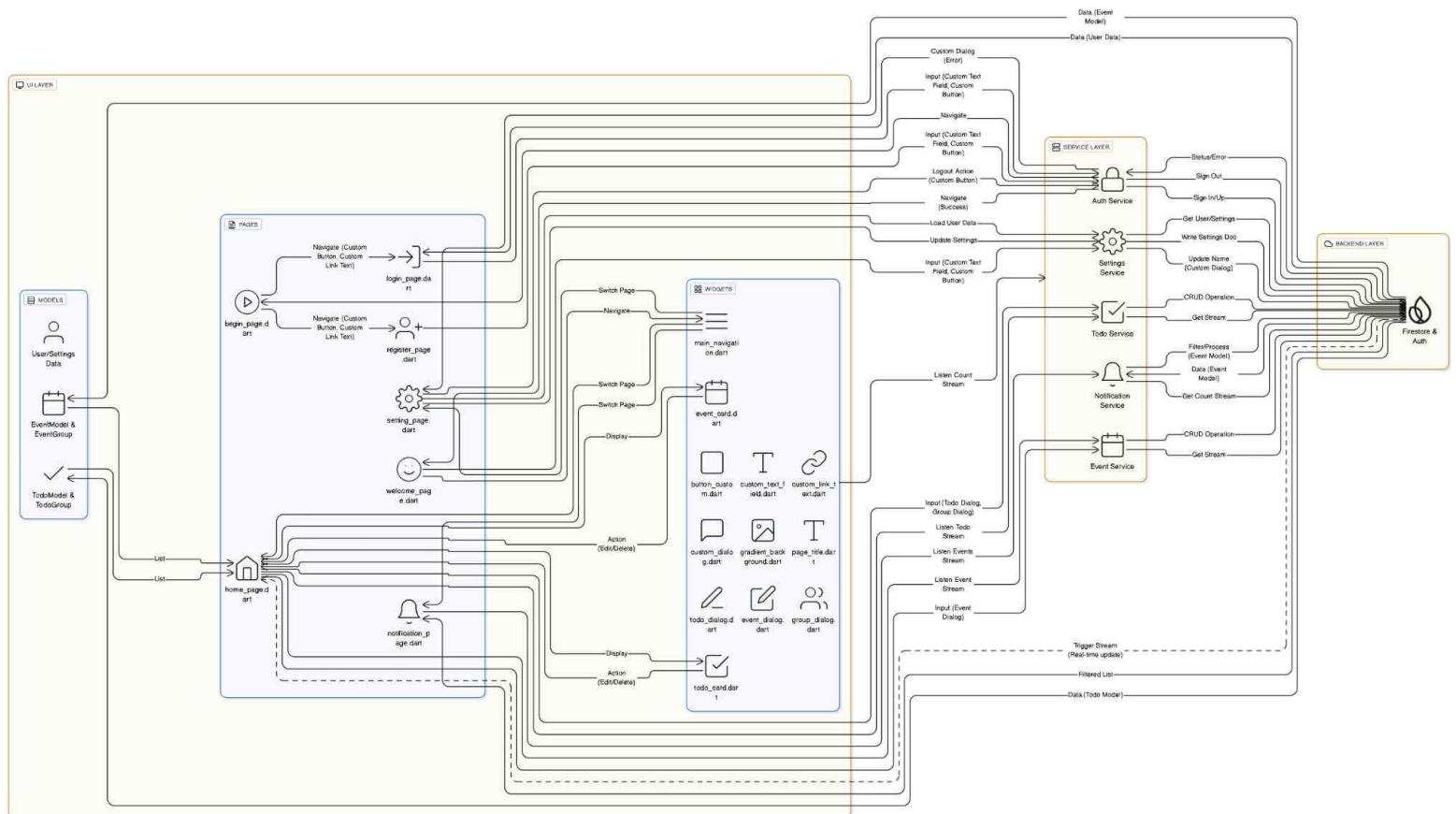
2. Packages และ Web Services ที่ใช้ (Dependencies & Backend)

แอปพลิเคชันนี้พึ่งพาบริการภายนอกที่สำคัญ ดังนี้

บริการ	แพ็กเกจ	ไฟล์ที่เกี่ยวข้อง	หน้าที่การทำงาน
Firebase Authentication	firebase_auth	auth_service.dart	จัดการการ Login, Register, Sign Out และการควบคุมผู้ใช้ทั้งหมด
Cloud Firestore	cloud_firestore	ไฟล์ Services ทั้งหมด	เป็นฐานข้อมูลหลักในการจัดเก็บข้อมูล Todo, Event, Group, User Profile และ Settings โดยรองรับการดึงข้อมูลแบบ Real-time (Stream)
External API	http, dart:convert , https://corsproxy.io/ ? https://zenquotes.io/api/today	home_page.dart	ตีงข้อความสร้างแรงบันดาลใจ (Quote of the Day) จาก zenquotes.io มาแสดงในหน้า Home

3. การเชื่อมโยงและการทำงาน (Interconnection and Data Flow)

การทำงานของแอปพลิเคชันเป็นไปตามลำดับขั้นตอนที่ชัดเจน โดย Services ทำหน้าที่เป็นสะพานเชื่อมระหว่าง Pages (UI) กับ Backend (Firestore)



4. โค้ดและคำอธิบาย (Code & Comments)

4.1 กลุ่ม Models (โครงสร้างข้อมูล)

4.1.1 event_model.dart

```
1. import 'package:cloud_firestore/cloud_firestore.dart';
2. import 'package:flutter/material.dart';
3.
4. // คลาส EventModel: ใช้เป็นโครงสร้างข้อมูลสำหรับ Event ที่จัดเก็บใน Firestore
5. // ทำหน้าที่แปลงข้อมูลระหว่าง Dart Object (EventModel) และ Firestore Document
6. class EventModel {
7.   final String id; //ID ของ Document ใน Firestore สำหรับ Event นี้
8.   final String userId; //ID ของผู้ใช้
9.   final String title; // ชื่อหรือหัวข้อของ Event
10.  final DateTime deadline; // Deadline ของ Event
11.  final Color borderColor; //สีขอบ/พื้นหลังของ Event Card (เก็บเป็นค่า Color ใน Flutter)
12.  final DateTime createdAt; // วันที่และเวลาที่ Event ถูกสร้างขึ้น
13.  final DateTime updatedAt; // วันที่และเวลาที่ Event ถูกอัปเดตล่าสุด
14.
15. // Constructor: ใช้สำหรับสร้าง EventModel object ภายใต้แนบฯ
16. EventModel({
17.   required this.id,
18.   required this.userId,
19.   required this.title,
20.   required this.deadline,
21.   required this.borderColor,
22.   required this.createdAt,
23.   required this.updatedAt,
24. });
25.
26. // Factory Constructor: ใช้สำหรับแปลง DocumentSnapshot (ข้อมูลดิบจาก Firestore) ให้เป็น EventModel object
27. factory EventModel.fromFirestore(DocumentSnapshot doc) {
28.   Map<String, dynamic> data = doc.data() as Map<String, dynamic>; // แปลงข้อมูลให้อยู่ในรูปแบบ Map
29.
30.   return EventModel(
31.     id: doc.id, // ตึง Document ID มาใช้เป็น Event ID
32.     userId: data['userId'] ?? '',
33.     title: data['title'] ?? '',
34.     deadline: (data['deadline'] as Timestamp?)?.toDate() ?? DateTime.now(), // แปลง Firestore Timestamp เป็น
35.     // Dart DateTime. ดำเนินการ ให้ใช้เวลาปัจจุบัน
36.     borderColor: data['borderColor'] != null // แปลงค่าสีที่เก็บเป็น int กลับมาเป็น Flutter Color
37.       ? Color(data['borderColor'])
38.       : Color.fromRGBO(255, 80, 79, 120), // กำหนดค่า default เป็นสี Dark Slate Blue
39.     createdAt: (data['createdAt'] as Timestamp?)?.toDate() ?? DateTime.now(),
40.     updatedAt: (data['updatedAt'] as Timestamp?)?.toDate() ?? DateTime.now(),
41.   );
42.
```

```

41.    }
42.
43.    // Method toMap(): ใช้สำหรับแปลง EventModel object กลับไปเป็น Map
44.    // เพื่อเตรียมบันทึก/อัปเดตข้อมูลใน Firestore
45.    Map<String, dynamic> toMap() {
46.
47.        return {
48.            'userId': userId,
49.            'title': title,
50.            'deadline': Timestamp.fromDate(deadline), // แปลง Dart DateTime กลับไปเป็น Firestore Timestamp
51.            'borderColor': borderColor.value, // แปลง Color เป็น int value เพื่อจัดเก็บใน Firestore
52.            'createdAt': Timestamp.fromDatecreatedAt(),
53.            'updatedAt': Timestamp.fromDateupdatedAt(),
54.        };
55.
56.        // Getter: ตรวจสอบว่า Event นี้เลยกำหนดเวลา (Deadline) ไปแล้วหรือยัง
57.        bool get isPast => deadline.isBefore(DateTime.now());
58.
59.        // Getter: ตัว数值จำนวนวันที่เหลือจนถึง Deadline (นับเฉพาะวัน ไม่รวมเวลา)
60.        int get daysUntilDeadline {
61.
62.            final now = DateTime.now();
63.            final today = DateTime(now.year, now.month, now.day);
64.            final deadlineDate = DateTime(deadline.year, deadline.month, deadline.day);
65.            // คำนวณความแตกต่างของวันและส่งกลับเป็นจำนวนเต็ม
66.            return deadlineDate.difference(today).inDays;
67.
68.        }
69.
70.        // Getter: ส่งกลับ Deadline ในรูปแบบ String ที่ถูกจัดฟอร์แมตแล้ว
71.        String get formattedDeadline {
72.
73.            // ใช้ฟังก์ชัน _getMonthName ในการแสดงชื่อเดือนย่อ
74.            return '${deadline.day} ${_getMonthName(deadline.month)} ${deadline.year}';
75.
76.        }
77.
78.        // Helper Method: แปลงหมายเลขเดือน (1-12) เป็นชื่อเดือนย่อ (Jan-Dec)
79.        String _getMonthName(int month) {
80.
81.            const months = [
82.                'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
83.                'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'
84.            ];
85.            return months[month - 1]; // month - 1 เพราะ List เริ่มจาก index 0
86.        }
87.    }

```

4.1.2 todo_model.dart

```
1. import 'package:cloud_firestore/cloud_firestore.dart';
2. // คลาส TodoModel: ใช้เป็นโครงสร้างข้อมูลสำหรับรายการ To-Do แต่ละรายการ
3. // ทำหน้าที่แปลงข้อมูลระหว่าง Dart Object (TodoModel) และ Firestore Document
4. class TodoModel {
5.
6.   final String id; // ID ของ Document ใน Firestore สำหรับ To-Do นี้
7.   final String userId; // ID ของผู้ใช้ที่
8.   final String title; // ชื่อหรือหัวข้อของ To-Do
9.   final String description; // คำอธิบายรายละเอียดเพิ่มเติม (มีค่าเริ่มต้นเป็นสตริงว่าง)
10.  final String emoji; // สีไมจิล่าหรับแสดงผล
11.  final bool isCompleted; // สถานะการเสร็จสมบูรณ์
12.  final int order; // ลำดับการจัดเรียงรายการ
13.  final String? groupId; // ID ของกลุ่มที่ To-Do นี้อยู่ (อาจเป็น null หากไม่มีกลุ่ม)
14.  final DateTime createdAt; // วันที่และเวลาที่ To-Do ถูกสร้างขึ้น
15.  final DateTime updatedAt; // วันที่และเวลาที่ To-Do ถูกอัปเดตล่าสุด
16.
17. // Constructor: ใช้สำหรับสร้าง TodoModel object ภายในแอปพลิเคชัน
18. TodoModel({
19.   required this.id,
20.   required this.userId,
21.   required this.title,
22.   this.description = '',
23.   this.emoji = '📝', // มีค่าเริ่มต้นเป็น '📝'
24.   this.isCompleted = false, // มีค่าเริ่มต้นเป็น false
25.   required this.order,
26.   this.groupId,
27.   required this.createdAt,
28.   required this.updatedAt,
29. });
30.
31. // Factory Constructor: ใช้สำหรับแปลง DocumentSnapshot (ข้อมูลจาก Firestore)
32. // ให้เป็น TodoModel object
33. factory TodoModel.fromFirestore(DocumentSnapshot doc) {
34.   Map<String, dynamic> data = doc.data() as Map<String, dynamic>;
35.   return TodoModel(
36.     id: doc.id, // ตั้ง Document ID มาใช้เป็น To-Do ID
37.     userId: data['userId'] ?? '',
38.     title: data['title'] ?? '',
39.     description: data['description'] ?? '',
40.     emoji: data['emoji'] ?? '📝',
41.     isCompleted: data['isCompleted'] ?? false,
42.     order: data['order'] ?? 0,
43.     groupId: data['groupId'], // ค่าเหล่านี้เป็น optional จึงไม่ต้องมีค่า default (null ได้)
44.     // แปลง Firestore Timestamp เป็น Dart DateTime
```

```

45.     createdAt: (data['createdAt'] as Timestamp?)?.toDate() ?? DateTime.now(),
46.     updatedAt: (data['updatedAt'] as Timestamp?)?.toDate() ?? DateTime.now(),
47.   );
48. }
49.
50. // Method toMap(): ใช้สำหรับแปลง TodoModel object กลับไปเป็น Map
51. // เพื่อเตรียมบันทึก/อัปเดตข้อมูลใน Firestore
52. Map<String, dynamic> toMap() {
53.   return {
54.     'userId': userId,
55.     'title': title,
56.     'description': description,
57.     'emoji': emoji,
58.     'isCompleted': isCompleted,
59.     'order': order,
60.     'groupId': groupId,
61.     // แปลง Dart DateTime กลับไปเป็น Firestore Timestamp
62.     'createdAt': Timestamp.fromDate(createdAt),
63.     'updatedAt': Timestamp.fromDate(updatedAt),
64.   };
65. }
66. }
67.
68. // คลาส TodoGroup: ใช้เป็นโครงสร้างข้อมูลสำหรับกลุ่ม (Category) ของ To-Do
69. class TodoGroup {
70.   final String id; // ID ของ Document ใน Firestore สำหรับกลุ่มนี้
71.   final String userId; // ID ของผู้ใช้
72.   final String name; // ชื่อของกลุ่ม To-Do
73.   final String emoji; // ลักษณะสำหรับแสดงผลลัพธ์
74.   final int order; // ลำดับการจัดเรียงกลุ่ม
75.   final DateTime createdAt; // วันที่และเวลาที่กลุ่มถูกสร้างขึ้น
76.
77. // Constructor: ใช้สำหรับสร้าง TodoGroup object
78. TodoGroup({
79.   required this.id,
80.   required this.userId,
81.   required this.name,
82.   this.emoji = '📝', // มีค่าเริ่มต้นเป็น '📝'
83.   required this.order,
84.   required this.createdAt,
85. });
86.
87. // Factory Constructor: ใช้สำหรับแปลง DocumentSnapshot ของกลุ่มจาก Firestore
88. // ให้เป็น TodoGroup object
89. factory TodoGroup.fromFirestore(DocumentSnapshot doc) {

```

```

90.     Map<String, dynamic> data = doc.data() as Map<String, dynamic>;
91.     return TodoGroup(
92.         id: doc.id, // ตั้ง Document ID มาใช้เป็น Group ID
93.         userId: data['userId'] ?? '',
94.         name: data['name'] ?? '',
95.         emoji: data['emoji'] ?? '❑',
96.         order: data['order'] ?? 0,
97.         // แปลง Firestore Timestamp เป็น Dart DateTime
98.         createdAt: (data['createdAt'] as Timestamp?)?.toDate() ?? DateTime.now(),
99.     );
100. }
101.
102. // Method toMap(): ใช้งานรับแปลง TodoGroup object กลับไปเป็น Map
103. // เพื่อเตรียมบันทึก/อัปเดตข้อมูลใน Firestore
104. Map<String, dynamic> toMap() {
105.     return {
106.         'userId': userId,
107.         'name': name,
108.         'emoji': emoji,
109.         'order': order,
110.         // แปลง Dart DateTime กลับไปเป็น Firestore Timestamp
111.         'createdAt': Timestamp.fromDate(createdAt),
112.     };
113. }

```

4.1.3 model.dart

```

1. // Export all models for easy importing
2. export 'todo_model.dart';
3. export 'event_model.dart';

```

4.2 กลุ่ม Services (ตระกูลและการติดต่อ Backend)

4.2.1 auth_service.dart

```
1. import 'package:firebase_auth/firebase_auth.dart';
2. import 'package:cloud_firestore/cloud_firestore.dart';
3.
4. // คลาส AuthService: จัดการการยืนยันตัวตน (Authentication) และข้อมูลผู้ใช้ใน Firestore
5. class AuthService {
6.   static final FirebaseAuth _auth = FirebaseAuth.instance; // Instance ของ Firebase Auth สำหรับจัดการการยืนยันตัวตน
7.   static final FirebaseFirestore _firestore = FirebaseFirestore.instance; // Instance ของ Firebase Firestore
8.   // สำหรับจัดการข้อมูลผู้ใช้
9.   static User? get currentUser => _auth.currentUser; // Get current user (Getter: ส่งคืน User object ของผู้ใช้ที่กำลังเข้าสู่ระบบ)
10.
11. // Sign in with email and password (เข้าสู่ระบบด้วยอีเมลและรหัสผ่าน)
12. static Future<UserCredential> signInWithEmailAndPassword({
13.   required String email,
14.   required String password,
15. }) async {
16.   return await _auth.signInWithEmailAndPassword(
17.     email: email.trim(), // ลบช่องว่างทั้งหมดของอีเมลก่อนส่ง
18.     password: password.trim(), // ลบช่องว่างทั้งหมดของรหัสผ่านก่อนส่ง
19.   );
20. }
21.
22. // Create user with email and password (สร้างบัญชีผู้ใช้ใหม่ด้วยอีเมลและรหัสผ่าน)
23. static Future<UserCredential> createUserWithEmailAndPassword({
24.   required String email,
25.   required String password,
26. }) async {
27.   return await _auth.createUserWithEmailAndPassword(
28.     email: email.trim(), // ลบช่องว่างทั้งหมดของอีเมลก่อนส่ง
29.     password: password.trim(), // ลบช่องว่างทั้งหมดของรหัสผ่านก่อนส่ง
30.   );
31. }
32.
33. // Sign out
34. static Future<void> signOut() async {
35.   await _auth.signOut();
36. }
37.
38. // Delete user account (ลบบัญชีผู้ใช้จาก Firebase Authentication)
39. static Future<void> deleteUser() async {
40.   final user = _auth.currentUser;
```

```
41.     if (user != null) {
42.         await user.delete(); // ลบบัญชีผู้ใช้ปัจจุบัน
43.     }
44. }
45.
46. // Update user display name (อัปเดตชื่อที่แสดงใน Firebase Auth Profile)
47. static Future<void> updateDisplayName(String name) async {
48.     final user = _auth.currentUser;
49.     if (user != null) {
50.         await user.updateDisplayName(name); // อัปเดตชื่อที่แสดง
51.     }
52. }
53.
54. // Create user document in Firestore (สร้าง Document ข้อมูลผู้ใช้ใน Collection 'users')
55. static Future<void> createUserDocument({
56.     required String uid,
57.     required String email,
58.     String? name,
59. }) async {
60.     await _firestore.collection('users').doc(uid).set({
61.         'email': email,
62.         'name': name,
63.         'createdAt': FieldValue.serverTimestamp(),
64.         'updatedAt': FieldValue.serverTimestamp(),
65.     });
66. }
67.
68. // Update user document in Firestore (อัปเดตข้อมูลผู้ใช้ใน Firestore)
69. static Future<void> updateUserDocument({
70.     required String uid,
71.     String? name,
72.     String? email,
73. }) async {
74.     Map<String, dynamic> data = {
75.         'updatedAt': FieldValue.serverTimestamp(), // อัปเดตเวลาการอัปเดต
76.     };
77.
78.     if (name != null) data['name'] = name; // เพิ่มชื่อถ้ามีการระบุ
79.     if (email != null) data['email'] = email; // เพิ่มอีเมลถ้ามีการระบุ
80.
81.     await _firestore.collection('users').doc(uid).set(
82.         data,
83.         SetOptions(merge: true), // ใช้ merge: true เพื่ออัปเดตเฉพาะ field ที่ระบุ
84.     );
85. }
```

```
86.
87. // Get user document from Firestore (ดึง Document ข้อมูลผู้ใช้จาก Firestore)
88. static Future<DocumentSnapshot> getUserDocument(String uid) async {
89.   return await _firestore.collection('users').doc(uid).get();
90. }
91.
92. // Check if user has name in Firestore (ตรวจสอบว่ามี field 'name' ใน Document หรือไม่)
93. static Future<bool> hasUserName(String uid) async {
94.   final doc = await getUserDocument(uid);
95.   // ตรวจสอบว่า Document มีอยู่จริง และ field 'name' มีค่าที่ไม่ใช่ null
96.   return doc.exists && (doc.data() as Map<String, dynamic>?)?['name'] != null;
97. }
98.
99. // Delete user data from Firestore (ลบข้อมูลทั้งหมดของผู้ใช้ที่เก็บขึ้นจาก Firestore)
100. static Future<void> deleteUserData(String uid) async {
101.   // Delete todos
102.   final todosSnapshot = await _firestore
103.     .collection('todos')
104.     .where('userId', isEqualTo: uid)
105.     .get();
106.   for (var doc in todosSnapshot.docs) {
107.     await doc.reference.delete(); // ลบ To-Do ทุกรายการ
108.   }
109.
110. // Delete events
111. final eventsSnapshot = await _firestore
112.   .collection('events')
113.   .where('userId', isEqualTo: uid)
114.   .get();
115. for (var doc in eventsSnapshot.docs) {
116.   await doc.reference.delete(); // ลบ Event ทุกรายการ
117. }
118.
119. // Delete groups
120. final groupsSnapshot = await _firestore
121.   .collection('groups')
122.   .where('userId', isEqualTo: uid)
123.   .get();
124. for (var doc in groupsSnapshot.docs) {
125.   await doc.reference.delete(); // ลบ Group ทุกรายการ
126. }
127.
128. // Delete settings
129. await _firestore.collection('settings').doc(uid).delete(); // ลบ Document การตั้งค่า
130.
```

```

131.     // Delete user data
132.     await _firestore.collection('users').doc(uid).delete(); // ลบ Document ข้อมูลผู้ใช้หลัก
133.   }
134.
135. // Get authentication error message (ແປງຮັກສ້ອຜິດພາດຂອງ Firebase Auth ເປັນຂ້ອຄວາມທີ່ເຫັນໄຈງາຍ)
136. static String getAuthErrorMessage(FirebaseAuthException e) {
137.   switch (e.code) {
138.     case 'invalid-credential':
139.       return 'Invalid email or password.';
140.     case 'invalid-email':
141.       return 'Please enter a valid email.';
142.     case 'user-disabled':
143.       return 'This account has been disabled.';
144.     case 'too-many-requests':
145.       return 'Too many attempts. Please try again later.';
146.     case 'user-not-found':
147.       return 'No account found for that email.';
148.     case 'wrong-password':
149.       return 'Incorrect password.';
150.     case 'email-already-in-use':
151.       return 'This email is already in use.';
152.     case 'weak-password':
153.       return 'Password is too weak.';
154.     case 'operation-not-allowed':
155.       return 'Email/password accounts are not enabled.';
156.   default:
157.     return e.message ?? 'An error occurred. Please try again.';
158.   }
159. }
160. }

```

4.2.2 event_service.dart

```

1. import 'package:cloud_firestore/cloud_firestore.dart';
2. import 'package:firebase_auth/firebase_auth.dart';
3. import 'package:flutter/material.dart';
4. import '../Model/event_model.dart';
5.
6. // คลาส EventService: ຈັດກາກາດຕັ້ງຂໍ້ມູນ, ກາຣເພີ່ມ, ວັບແດດ, ແລະລົບ Events ໃນ Firestore
7. class EventService {
8.   static final FirebaseFirestore _firestore = FirebaseFirestore.instance; // Instance ຂອງ Firestore ສໍາໜັບຕົດຕ່ອກກົນ
   ຖ້ານຂໍ້ມູນ
9.   static final FirebaseAuth _auth = FirebaseAuth.instance; // Instance ຂອງ FirebaseAuth ສໍາໜັບຕົດຕ່ອກກົນ
   ປູ້ໃຊ້
10.
11. // Get all events for current user (ດຶງຮາຍການ Event ທັງໝາຍຂອງຜູ້ໃຊ້ປໍ່ຈຸບັນແນວ Stream)

```

```
12.     static Stream<List<EventModel>> getEventsStream() {
13.         final user = _auth.currentUser;
14.         // តាមពីរដែលបានក្រោមនេះ នឹងត្រូវបាន Stream រាយការណ៍
15.         if (user == null) return Stream.value([]);
16.
17.         return _firestore
18.             .collection('events') // ការណុទ Collection
19.             .where('userId', isEqualTo: user.uid) // ក្រឡែខាងក្រោម Events នៃអ្នកដ្ឋាននេះ
20.             .snapshots() // តើងខ្សោយនូវនៅក្នុង Real-time Stream
21.             .map(
22.                 (snapshot) => snapshot
23.                     .docs // បញ្ចប់ QuerySnapshot ដើម្បីជួយ List<EventModel>
24.                     .map(
25.                         (doc) => EventModel.fromFirestore(doc),
26.                     ) // បញ្ចប់ព័ត៌មាន Document ដើម្បីជួយ EventModel
27.                     .toList(),
28.             ); // បញ្ចប់ជួយ List
29.     }
30.
31.     // Add new event (ធ្វើការថាមពេលនៃ Firestore)
32.     static Future<String> addEvent({
33.         required String title,
34.         required DateTime deadline,
35.         required Color borderColor,
36.     }) async {
37.         final user = _auth.currentUser;
38.         // ត្រូវសិក្សាការបញ្ជីនៅក្នុងគណន៍
39.         if (user == null) throw Exception('User not authenticated');
40.
41.         // ធ្វើការថាមពេលនៃ Collection 'events'
42.         final docRef = await _firestore.collection('events').add({
43.             'userId': user.uid, // ឈ្មោះការថាមពេលនៃ User ID
44.             'title': title,
45.             'deadline': Timestamp.fromDate(
46.                 deadline,
47.             ), // បញ្ចប់ DateTime ដើម្បីជួយ Firestore Timestamp
48.             'borderColor': borderColor.value, // កើតគ្នាតាមពេល integer
49.             'createdAt': FieldValue.serverTimestamp(), // ពេលវេលាដំឡើង
50.             'updatedAt': FieldValue.serverTimestamp(), // ពេលវេលាដំឡើង
51.         });
52.
53.         return docRef.id; // សេចក្តីនៃការថាមពេលនៃការបញ្ចប់ថា ត្រូវបានក្រោមនេះ
54.     }
55.
56.     // Update existing event (អំពើការថាមពេលនៃការបញ្ចប់ដែលត្រូវក្រោមនេះ)
```

```
57.     static Future<void> updateEvent({
58.         required String eventId,
59.         required String title,
60.         required DateTime deadline,
61.         required Color borderColor,
62.     }) async {
63.         // อัปเดต Document โดยใช้ eventId
64.         await _firestore.collection('events').doc(eventId).update({
65.             'title': title,
66.             'deadline': Timestamp.fromDate(deadline), // อัปเดตกำหนดเวลา
67.             'borderColor': borderColor.value, // อัปเดตสี
68.             'updatedAt': FieldValue.serverTimestamp(), // อัปเดตเวลาล่าสุด
69.         });
70.     }
71.
72.     // Delete event (ลบ Event ออกจาก Firestore)
73.     static Future<void> deleteEvent(String eventId) async {
74.         await _firestore.collection('events').doc(eventId).delete(); // ลบ Document โดยใช้ eventId
75.     }
76.
77.     // Get event by ID (ดึงข้อมูล Event เฉพาะรายการด้วย ID)
78.     static Future<EventModel?> getEventById(String eventId) async {
79.         final doc = await _firestore.collection('events').doc(eventId).get();
80.         // ตรวจสอบว่า Document มีอยู่จริง
81.         if (doc.exists) {
82.             return EventModel.fromFirestore(doc); // แปลง Document เป็น EventModel
83.         }
84.         return null; // คืนค่า null หากไม่พบ
85.     }
86.
87.     // Get events by date range (ดึงรายการ Events ภายในช่วงวันที่ที่กำหนด)
88.     static Future<List<EventModel>> getEventsByDateRange({
89.         required DateTime startDate,
90.         required DateTime endDate,
91.     }) async {
92.         final user = _auth.currentUser;
93.         // ถ้าไม่มีผู้ใช้ ให้คืนค่า List ว่าง
94.         if (user == null) return [];
95.
96.         // ดึงข้อมูลแบบ Query โดยมีเงื่อนไข
97.         final snapshot = await _firestore
98.             .collection('events')
99.             .where('userId', isEqualTo: user.uid) // กรองเฉพาะผู้ใช้ปัจจุบัน
100.            .where(
101.                'deadline',
```

```

102.        isGreaterThanOrEqualTo: Timestamp.fromDate(startDate),
103.    ) // กำหนดเวลาต้องมากกว่าหรือเท่ากับวันเริ่มต้น
104.    .where(
105.        'deadline',
106.        isLessThanOrEqualTo: Timestamp.fromDate(endDate),
107.    ) // กำหนดเวลาต้องห้อยกว่าหรือเท่ากับวันสิ้นสุด
108.    .get(); // ดึงข้อมูลแบบครั้งเดียว (Future)
109.
110.    // แปลง QuerySnapshot เป็น List<EventModel>
111.    return snapshot.docs.map((doc) => EventModel.fromFirestore(doc)).toList();
112. }
113.
114.

```

4.2.3 todo_service.dart

```

1. import 'package:cloud_firestore/cloud_firestore.dart'; // Import สำหรับ Firebase Firestore
2. import 'package:firebase_auth/firebase_auth.dart'; // Import สำหรับ Firebase Authentication
3. import '../Model/todo_model.dart'; // Import โมเดลข้อมูล Todo และ TodoGroup
4.
5. class TodoService {
6.     // คลาสบริการจัดการข้อมูล Todo และ TodoGroup ใน Firestore
7.     static final FirebaseFirestore _firestore =
8.         FirebaseFirestore.instance; // Instance ของ Firestore
9.     static final FirebaseAuth _auth =
10.        FirebaseAuth.instance; // Instance ของ FirebaseAuth
11.
12.    // Get all todos for current user (ดึงรายการ Todo หลักทั้งหมดของผู้ใช้ปัจจุบันแบบ Stream)
13.    static Stream<List<TodoModel>> getTodosStream() {
14.        final user = _auth.currentUser; // ดึงผู้ใช้ปัจจุบัน
15.        if (user == null)
16.            return Stream.value([]); // ถ้าไม่มีผู้ใช้ ให้คืนค่า Stream ว่าง
17.
18.        return _firestore
19.            .collection('todos') // เลือก Collection 'todos'
20.            .where('userId', isEqualTo: user.uid) // กรองเฉพาะ Todo ของผู้ใช้คนนี้
21.            .where(
22.                'parentId',
23.                isEqualTo: null,
24.            ) // กรองเฉพาะ Todo หลัก (ไม่เป็น Sub-task)
25.            .snapshots() // ดึงข้อมูลแบบ Real-time Stream
26.            .map(
27.                (snapshot) => snapshot
28.                    .docs // แปลง QuerySnapshot
29.                    .map(

```

```

30.             (doc) => TodoModel.fromFirestore(doc),
31.         ) // แปลงแต่ละ Document เป็น TodoModel
32.         .toList(),
33.     ); // แปลงเป็น List
34. }
35.
36. // Get all groups for current user (ดึงรายการ Group ทั้งหมดของผู้ใช้ปัจจุบันแบบ Stream)
37. static Stream<List<TodoGroup>> getGroupsStream() {
38.     final user = _auth.currentUser; // ดึงผู้ใช้ปัจจุบัน
39.     if (user == null)
40.         return Stream.value([]); // ถ้าไม่มีผู้ใช้ ให้คืนค่า Stream ว่าง
41.
42.     return _firestore
43.         .collection('groups') // เลือก Collection 'groups'
44.         .where('userId', isEqualTo: user.uid) // กรองเฉพาะ Group ของผู้ใช้คนนี้
45.         .snapshots() // ดึงข้อมูลแบบ Real-time Stream
46.         .map(
47.             (snapshot) => snapshot
48.                 .docs // แปลง QuerySnapshot
49.                 .map(
50.                     (doc) => TodoGroup.fromFirestore(doc),
51.                 ) // แปลงแต่ละ Document เป็น TodoGroup
52.                 .toList(),
53.             ); // แปลงเป็น List
54. }
55.
56. // Add new todo (เพิ่ม Todo ใหม่)
57. static Future<String> addTodo({
58.     required String title,
59.     String description = '',
60.     String emoji = '📝',
61.     String? groupId,
62. }) async {
63.     final user = _auth.currentUser; // ดึงผู้ใช้ปัจจุบัน
64.     if (user == null)
65.         throw Exception('User not authenticated'); // ตรวจสอบการยืนยันตัวตน
66.
67.     // Get max order (หาค่า 'order' สูงสุดที่มีอยู่)
68.     final todosSnapshot = await _firestore
69.         .collection('todos') // เลือก Collection 'todos'
70.         .where('userId', isEqualTo: user.uid) // กรองเฉพาะของผู้ใช้
71.         .get(); // ดึงข้อมูลแบบครั้งเดียว
72.
73.     int maxOrder = 0; // กำหนดค่าเริ่มต้น
74.     for (var doc in todosSnapshot.docs) {

```

```
75.      // วนลูปผ่านเอกสารทั้งหมด
76.      final order = doc.data()['order'] ?? 0; // ดึงค่า 'order'
77.      if (order > maxOrder) maxOrder = order; // อัปเดต maxOrder
78.  }
79.
80.  final docRef = await _firestore.collection('todos').add({
81.    // เพิ่ม Document ใหม่
82.    'userId': user.uid, // ผูก Todo เข้ากับ User ID
83.    'title': title,
84.    'description': description,
85.    'emoji': emoji,
86.    'isCompleted': false, // ค่าเริ่มต้นเป็นยังไงไม่เสร็จ
87.    'order': maxOrder + 1, // กำหนด 'order' เป็นค่าสูงสุด + 1
88.    'groupId': groupId, // Group ID (อาจเป็น null)
89.    'parentId': null, // Todo หลัก ไม่มี Parent ID
90.    'createdAt': FieldValue.serverTimestamp(), // ประทับเวลาสร้าง
91.    'updatedAt': FieldValue.serverTimestamp(), // ประทับเวลาอัปเดต
92.  });
93.
94.  return docRef.id; // ส่งคืน ID ของ Document ที่สร้างขึ้นใหม่
95. }
96.
97. // Update existing todo (อัปเดตข้อมูล Todo ที่มีอยู่)
98. static Future<void> updateTodo({
99.   required String todoId,
100.  required String title,
101.  String description = '',
102.  String emoji = '☑',
103.  String? groupId,
104. }) async {
105.   await _firestore.collection('todos').doc(todoId).update({
106.     // อัปเดต Document โดยใช้ todoId
107.     'title': title,
108.     'description': description,
109.     'emoji': emoji,
110.     'groupId': groupId, // อัปเดต Group ID
111.     'updatedAt': FieldValue.serverTimestamp(), // อัปเดตเวลาล่าสุด
112.   });
113. }
114.
115. // Toggle todo completion (สลับสถานะการเสร็จสิ้นของ Todo)
116. static Future<void> toggleTodoComplete(
117.   String todoId,
118.   bool isCompleted,
119. ) async {
```

```

120.     await _firestore.collection('todos').doc(todoId).update({
121.         // อัปเดต Document
122.         'isCompleted': !isCompleted, // สลับค่า isCompleted
123.         'updatedAt': FieldValue.serverTimestamp(), // อัปเดตเวลาล่าสุด
124.     });
125. }
126.
127. // Delete todo (ลบ Todo ออกจาก Firestore)
128. static Future<void> deleteTodo(String todoId) async {
129.     await _firestore
130.         .collection('todos')
131.         .doc(todoId)
132.         .delete(); // ลบ Document โดยใช้ todoId
133. }
134.
135. // Reorder todos (จัดเรียงลำดับ 'order' ของ Todo ใหม่)
136. static Future<void> reorderTodos(List<TodoModel> todos) async {
137.     final batch = _firestore
138.         .batch(); // สั่ง Write Batch เพื่อดำเนินการหลายอย่างพร้อมกัน
139.
140.     for (int i = 0; i < todos.length; i++) {
141.         // วนลูปผ่านรายการ Todo ที่จัดเรียงแล้ว
142.         final docRef = _firestore
143.             .collection('todos')
144.             .doc(todos[i].id); // อ้างอิงถึง Document
145.         batch.update(docRef, {
146.             // เพิ่มการอัปเดตเข้าใน Batch
147.             'order':
148.                 todos.length -
149.                     i, // กำหนด 'order' ใหม่ (ใช้ index แบบกลับด้านเพื่อให้ตัวแรกมี order สูงสุด)
150.             'updatedAt': FieldValue.serverTimestamp(), // อัปเดตเวลาล่าสุด
151.         });
152.     }
153.
154.     await batch.commit(); // ยืนยันการดำเนินการทั้งหมดใน Batch
155. }
156.
157. // Add new group (เพิ่ม Group ใหม่)
158. static Future<String> addGroup({
159.     required String name,
160.     String emoji = '❑',
161. }) async {
162.     final user = _auth.currentUser; // ดึงผู้ใช้ปัจจุบัน
163.     if (user == null)
164.         throw Exception('User not authenticated'); // ตรวจสอบการยืนยันตัวตน

```

```

165.
166.    // Get max order (หาค่า 'order' สูงสุดของ Group ที่มีอยู่)
167.    final groupsSnapshot = await _firestore
168.        .collection('groups') // เลือก Collection 'groups'
169.        .where('userId', isEqualTo: user.uid) // กรองเฉพาะของผู้ใช้
170.        .get(); // ดึงข้อมูลแบบครั้งเดียว
171.
172.    int maxOrder = 0; // กำหนดค่าเริ่มต้น
173.    for (var doc in groupsSnapshot.docs) {
174.        // วนลูปผ่านเอกสารทั้งหมด
175.        final order = doc.data()['order'] ?? 0; // ตั้งค่า 'order'
176.        if (order > maxOrder) maxOrder = order; // อัปเดต maxOrder
177.    }
178.
179.    final docRef = await _firestore.collection('groups').add({
180.        // เพิ่ม Document Group ใหม่
181.        'userId': user.uid, // ผูก Group เข้ากับ User ID
182.        'name': name,
183.        'emoji': emoji,
184.        'order': maxOrder + 1, // กำหนด 'order' เป็นค่าสูงสุด + 1
185.        'createdAt': FieldValue.serverTimestamp(), // ประทับเวลาสร้าง
186.    });
187.
188.    return docRef.id; // ส่งคืน ID ของ Document Group ที่สร้างขึ้นใหม่
189. }
190.
191. // Delete group (ลบ Group)
192. static Future<void> deleteGroup(String groupId) async {
193.     // First, move all todos in this group to no group (อันดับแรก ย้าย todos ทั้งหมดใน group นี้ออก)
194.     final todosSnapshot = await _firestore
195.         .collection('todos') // เลือก Collection 'todos'
196.         .where('groupId', isEqualTo: groupId) // กรอง Todo ที่อยู่ใน Group นี้
197.         .get(); // ดึงข้อมูลแบบครั้งเดียว
198.
199.     final batch = _firestore.batch(); // สร้าง Write Batch
200.     for (var doc in todosSnapshot.docs) {
201.         // วนลูปผ่าน Todo ที่เกี่ยวข้อง
202.         batch.update(doc.reference, {
203.             'groupId': null,
204.         }); // อัปเดต 'groupId' ให้เป็น null
205.     }
206.
207.     // Then delete the group (จากนั้นลบ Group)
208.     batch.delete(
209.         _firestore.collection('groups').doc(groupId),

```

```

210.    ); // เพิ่มค่าส์ลั่น Group เข้าใน Batch
211.    await batch.commit(); // ยืนยันการดำเนินการทั้งหมดใน Batch
212.  }
213. }

```

4.2.4 notification_service.dart

```

1. import 'package:cloud_firestore/cloud_firestore.dart'; // Import สำหรับ Firebase Firestore
2. import 'package:firebase_auth/firebase_auth.dart'; // Import สำหรับ Firebase Authentication
3. import 'package:flutter/material.dart'; // Import สำหรับการใช้ Color และ IconData ของ Flutter
4.
5. class NotificationService {
6.   // คลาสบริการสำหรับดึงข้อมูลการแจ้งเตือนจาก Firestore
7.   static final FirebaseFirestore _firestore =
8.     FirebaseFirestore.instance; // Instance ของ Firestore
9.   static final FirebaseAuth _auth =
10.    FirebaseAuth.instance; // Instance ของ FirebaseAuth
11.
12. // Get notifications stream (ดึงรายการการแจ้งเตือนสำหรับ Event ที่จะถึงกำหนดวันนี้/พรุ่งนี้แบบ Stream)
13. static Stream<List<Map<String, dynamic>>> getNotificationsStream() {
14.   final user = _auth.currentUser; // ดึงผู้ใช้ปัจจุบัน
15.   if (user == null)
16.     return Stream.value([]); // ถ้าไม่มีผู้ใช้ ให้คืนค่า Stream ว่าง
17.
18.   return _firestore
19.     .collection('events') // เลือก Collection 'events'
20.     .where('userId', isEqualTo: user.uid) // กรองเฉพาะ Event ของผู้ใช้คนนี้
21.     .snapshots() // ดึงข้อมูลแบบ Real-time Stream
22.     .map((snapshot) {
23.       // แปลง QuerySnapshot เป็น List ของ Map
24.       final now = DateTime.now(); // เวลาปัจจุบัน
25.       final today = DateTime(
26.         now.year,
27.         now.month,
28.         now.day,
29.       ); // วันนี้ (เที่ยงคืน)
30.       final tomorrow = today.add(
31.         const Duration(days: 1),
32.       ); // 明天 (เที่ยงคืน)
33.
34.       List<Map<String, dynamic>> notifications =
35.         []; // สร้าง List เป็นสำหรับเก็บการแจ้งเตือน
36.
37.       for (var doc in snapshot.docs) {
38.         // วนลูปผ่านเอกสาร Event ทั้งหมด

```

```

39.         final data = doc.data(); // ดึงข้อมูลของเอกสาร
40.         final deadline = (data['deadline'] as Timestamp)
41.             .toDate(); // แปลง Timestamp เป็น DateTime
42.         final deadlineDay = DateTime(
43.             // วันที่ของ Deadline (ตัดเวลาออก)
44.             deadline.year,
45.             deadline.month,
46.             deadline.day,
47.         );
48.
49.         // Check if it's today or tomorrow (ตรวจสอบว่าเป็นวันนี้หรือพรุ่งนี้)
50.         if (deadlineDay == today) {
51.             // ถ้าเป็นวันนี้
52.             notifications.add({
53.                 // เพิ่มข้อมูลการแจ้งเตือนสำหรับวันนี้
54.                 'id': doc.id,
55.                 'title': data['title'],
56.                 'deadline': deadline,
57.                 'type': 'today',
58.                 'message': 'Event deadline is today!',
59.                 'borderColor': Color(
60.                     data['borderColor'],
61.                 ), // แปลง int เป็น Color
62.                 'icon': Icons.event_available,
63.                 'iconColor': Colors.red,
64.             });
65.         } else if (deadlineDay == tomorrow) {
66.             // ถ้าเป็นพรุ่งนี้
67.             notifications.add({
68.                 // เพิ่มข้อมูลการแจ้งเตือนสำหรับพรุ่งนี้
69.                 'id': doc.id,
70.                 'title': data['title'],
71.                 'deadline': deadline,
72.                 'type': 'tomorrow',
73.                 'message': 'Event deadline is tomorrow!',
74.                 'borderColor': Color(
75.                     data['borderColor'],
76.                 ), // แปลง int เป็น Color
77.                 'icon': Icons.event_note,
78.                 'iconColor': Colors.orange,
79.             });
80.         }
81.     }
82.
83.     // Sort by type (today first) then by deadline (เรียงลำดับ: 'today' มาก่อน, จากนั้นเรียงตามเวลา Deadline)

```

```

84.         notifications.sort((a, b) {
85.             if (a['type'] == 'today' && b['type'] != 'today')
86.                 return -1; // 'today' มาก่อน
87.             if (a['type'] != 'today' && b['type'] == 'today')
88.                 return 1; // 'today' มาหลัง
89.             return (a['deadline'] as DateTime).compareTo(
90.                 b['deadline'] as DateTime,
91.             ); // เรียงตามเวลา Deadline
92.         });
93.
94.         return notifications; // ส่งคืนรายการการแจ้งเตือนที่กรองและเรียงลำดับแล้ว
95.     });
96. }
97.
98. // Get notification count (ตั้งจำนวนการแจ้งเตือนแบบ Future/ครั้งเดียว)
99. static Future<int> getNotificationCount() async {
100.     final user = _auth.currentUser; // ล็อกอินปัจจุบัน
101.     if (user == null) return 0; // ถ้าไม่มีผู้ใช้ คืนค่า 0
102.
103.     try {
104.         final now = DateTime.now(); // เวลาปัจจุบัน
105.         final today = DateTime(
106.             now.year,
107.             now.month,
108.             now.day,
109.         ); // วันนี้ (เที่ยงคืน)
110.         final tomorrow = today.add(
111.             const Duration(days: 1),
112.         ); // พุ่งนี้ (เที่ยงคืน)
113.
114.         final eventsSnapshot = await _firestore
115.             .collection('events') // เลือก Collection 'events'
116.             .where(
117.                 'userId',
118.                 isEqualTo: user.uid,
119.             ) // กรองเฉพาะ Event ของผู้ใช้คนนี้
120.             .get(); // ตั้งข้อมูลแบบครั้งเดียว
121.
122.         int count = 0; // ตัวนับ
123.         for (var doc in eventsSnapshot.docs) {
124.             // วนลูปผ่านเอกสาร Event ทั้งหมด
125.             final data = doc.data(); // ดึงข้อมูลของเอกสาร
126.             final deadline = (data['deadline'] as Timestamp)
127.                 .toDate(); // แปลง Timestamp เป็น DateTime
128.             final deadlineDay = DateTime(

```

```

129.         // วันที่ของ Deadline (ตัดเวลาออก)
130.         deadline.year,
131.         deadline.month,
132.         deadline.day,
133.     );
134.
135.     // Count only today and tomorrow (นับเฉพาะ Event ที่จะถึงกำหนดวันนี้หรือพรุ่งนี้)
136.     if (deadlineDay == today || deadlineDay == tomorrow) {
137.         count++; // เพิ่มจำนวน
138.     }
139. }
140.
141. return count; // ส่งคืนจำนวนการแจ้งเตือน
142. } catch (e) {
143.     // ตักจับข้อผิดพลาด
144.     print('Error getting notification count: $e'); // พิมพ์ข้อผิดพลาด
145.     return 0; // คืนค่า 0 เมื่อเกิดข้อผิดพลาด
146. }
147. }
148.
149. // Get notification count stream (ดึงจำนวนการแจ้งเตือนแบบ Real-time Stream)
150. static Stream<int> getNotificationCountStream() {
151.     final user = _auth.currentUser; // ดึงผู้ใช้ปัจจุบัน
152.     if (user == null) return Stream.value(0); // ถ้าไม่มีผู้ใช้ คืนค่า Stream 0
153.
154.     return _firestore
155.         .collection('events') // เลือก Collection 'events'
156.         .where('userId', isEqualTo: user.uid) // กรองเฉพาะ Event ของผู้ใช้คนนี้
157.         .snapshots() // ดึงข้อมูลแบบ Real-time Stream
158.         .map((snapshot) {
159.             // แปลง QuerySnapshot เป็นจำนวนนับ
160.             final now = DateTime.now(); // เวลาปัจจุบัน
161.             final today = DateTime(
162.                 now.year,
163.                 now.month,
164.                 now.day,
165.             ); // วันนี้ (เทียบศูนย์)
166.             final tomorrow = today.add(
167.                 const Duration(days: 1),
168.             ); // 明日 (เทียบศูนย์)
169.
170.             int count = 0; // ตัวนับ
171.             for (var doc in snapshot.docs) {
172.                 // วนลูปผ่านเอกสาร Event ทั้งหมด
173.                 final data = doc.data(); // ดึงข้อมูลของเอกสาร

```

```

174.         final deadline = (data['deadline'] as Timestamp)
175.             .toDate(); // แปลง Timestamp เป็น DateTime
176.         final deadlineDay = DateTime(
177.             // วันที่ของ Deadline (ตัดเวลาออก)
178.             deadline.year,
179.             deadline.month,
180.             deadline.day,
181.         );
182.
183.         // Count only today and tomorrow (นับเฉพาะ Event ที่จะถึงกำหนดวันนี้หรือพรุ่งนี้)
184.         if (deadlineDay == today || deadlineDay == tomorrow) {
185.             count++; // เพิ่มตัวนับ
186.         }
187.     }
188.
189.     return count; // ส่งคืนจำนวนการแจ้งเตือน
190. });
191. }
192. }
```

4.2.5 setting_service.dart

```

1. import 'package:cloud_firestore/cloud_firestore.dart'; // Import สำหรับ Firebase Firestore
2. import 'package:firebase_auth/firebase_auth.dart'; // Import สำหรับ Firebase Authentication
3.
4. class SettingsService {
5.     // คลาสบริการสำหรับจัดการข้อมูล Settings และ Profile ของผู้ใช้
6.     static final FirebaseFirestore _firestore =
7.         FirebaseFirestore.instance; // Instance ของ Firestore
8.     static final FirebaseAuth _auth =
9.         FirebaseAuth.instance; // Instance ของ FirebaseAuth
10.
11.    // Get user settings (ดึงข้อมูล Settings ของผู้ใช้แบบ Future/ครั้งเดียว)
12.    static Future<Map<String, dynamic>> getUserSettings() async {
13.        final user = _auth.currentUser; // ดึงผู้ใช้ปัจจุบัน
14.        if (user == null) return {}; // ถ้าไม่มีผู้ใช้ คืนค่า Map ว่าง
15.
16.        try {
17.            final doc = await _firestore
18.                .collection('settings') // เลือก Collection 'settings'
19.                .doc(user.uid) // ระบุเอกสารด้วย UID ของผู้ใช้
20.                .get(); // ดึงข้อมูลแบบครั้งเดียว
21.
22.            if (doc.exists) {
23.                // ถ้าเอกสารมีอยู่
```

```

24.         return doc.data() ?? {}; // คืนค่าข้อมูล หรือ Map ว่างหากข้อมูลเป็น null
25.     }
26.     return {}; // ถ้าเอกสารไม่มีอยู่ คืนค่า Map ว่าง
27. } catch (e) {
28.     // ตักจับข้อผิดพลาด
29.     print('Error getting user settings: $e'); // พิมพ์ข้อผิดพลาด
30.     return {}; // คืนค่า Map ว่างเมื่อเกิดข้อผิดพลาด
31. }
32. }

33.

34. // Get user settings stream (ดึงข้อมูล Settings ของผู้ใช้แบบ Real-time Stream)
35. static Stream<Map<String, dynamic>> getUserSettingsStream() {
36.     final user = _auth.currentUser; // ตั้งผู้ใช้ปัจจุบัน
37.     if (user == null)
38.         return Stream.value({}); // ถ้าไม่มีผู้ใช้ คืนค่า Stream Map ว่าง
39.
40.     return _firestore
41.         .collection('settings') // เลือก Collection 'settings'
42.         .doc(user.uid) // ระบุเอกสารด้วย UID ของผู้ใช้
43.         .snapshots() // ดึงข้อมูลแบบ Real-time Stream
44.         .map(
45.             (doc) => doc.data() ?? {}),
46.         ); // แปลง DocumentSnapshot เป็น Map หรือ Map ว่าง
47. }

48.

49. // Save user settings (บันทึกหรืออัปเดต Settings ของผู้ใช้)
50. static Future<void> saveUserSettings({
51.     bool? notifications, // พารามิเตอร์สำหรับเปิด/ปิดการแจ้งเตือน
52.     bool? sound, // พารามิเตอร์สำหรับเปิด/ปิดเสียง
53.     Map<String, dynamic>?
54.     additionalSettings, // พารามิเตอร์สำหรับ Settings เพิ่มเติม
55. }) async {
56.     final user = _auth.currentUser; // ตั้งผู้ใช้ปัจจุบัน
57.     if (user == null) return; // ถ้าไม่มีผู้ใช้ ไม่ต้องทำอะไร
58.
59.     try {
60.         Map<String, dynamic> settings = {
61.             'updatedAt': FieldValue.serverTimestamp(), // ประทับเวลาที่อัปเดต
62.         };
63.
64.         if (notifications != null)
65.             settings['notifications'] =
66.                 notifications; // เพิ่ม notifications ถ้ามีค่า
67.         if (sound != null) settings['sound'] = sound; // เพิ่ม sound ถ้ามีค่า
68.         if (additionalSettings != null) {

```

```

69.         settings.addAll(additionalSettings); // เพิ่ม settings อื่นๆ
70.     }
71.
72.     await _firestore
73.         .collection('settings') // เลือก Collection 'settings'
74.         .doc(user.uid) // ระบุเอกสารด้วย UID ของผู้ใช้
75.         .set(
76.             settings,
77.             SetOptions(merge: true),
78.         ); // บันทึกข้อมูล (Merge: true คืออัปเดตเฉพาะ field ที่กำหนด)
79.     } catch (e) {
80.         // ตัดจับข้อผิดพลาด
81.         print('Error saving user settings: $e'); // พิมพ์ข้อผิดพลาด
82.         rethrow; // โยน Exception ต่อไป
83.     }
84. }
85.
86. // Update notification setting (อัปเดตเฉพาะการตั้งค่าการแจ้งเตือน)
87. static Future<void> updateNotificationSetting(bool enabled) async {
88.     await saveUserSettings(notifications: enabled); // เรียกใช้ saveUserSettings
89. }
90.
91. // Update sound setting (อัปเดตเฉพาะการตั้งค่าเสียง)
92. static Future<void> updateSoundSetting(bool enabled) async {
93.     await saveUserSettings(sound: enabled); // เรียกใช้ saveUserSettings
94. }
95.
96. // Get user profile data (ดึงข้อมูล Profile ของผู้ใช้จาก Collection 'users')
97. static Future<Map<String, dynamic>> getUserProfile() async {
98.     final user = _auth.currentUser; // ดึงผู้ใช้ปัจจุบัน
99.     if (user == null) return {}; // ถ้าไม่มีผู้ใช้ คืนค่า Map ว่าง
100.
101.    try {
102.        final doc = await _firestore
103.            .collection('users') // เลือก Collection 'users'
104.            .doc(user.uid) // ระบุเอกสารด้วย UID ของผู้ใช้
105.            .get(); // ดึงข้อมูลแบบครึ่งเดียว
106.
107.        if (doc.exists) {
108.            // ถ้าเอกสารมีอยู่
109.            return doc.data() ?? {}; // คืนค่าข้อมูล หรือ Map ว่างหากข้อมูลเป็น null
110.        }
111.        return {}; // ถ้าเอกสารไม่มีอยู่ คืนค่า Map ว่าง
112.    } catch (e) {
113.        // ตัดจับข้อผิดพลาด

```

```

114.     print('Error getting user profile: $e'); // พิมพ์ข้อผิดพลาด
115.     return {}; // คืนค่า Map ว่างเมื่อเกิดข้อผิดพลาด
116.   }
117. }
118.
119. // Update user profile (อัปเดตข้อมูล Profile ของผู้ใช้)
120. static Future<void> updateUserProfile({
121.   String? name, // นามสกุล
122.   String? email, // อีเมล
123. }) async {
124.   final user = _auth.currentUser; // ดึงผู้ใช้ปัจจุบัน
125.   if (user == null) return; // ถ้าไม่มีผู้ใช้ ไม่ต้องทำอะไร
126.
127.   try {
128.     Map<String, dynamic> data = {
129.       'updatedAt': FieldValue.serverTimestamp(), // ประทับเวลาที่อัปเดต
130.     };
131.
132.     if (name != null) {
133.       data['name'] = name; // เพิ่ม name ใน Map
134.       // Also update Firebase Auth display name
135.       await user.updateDisplayName(name); // อัปเดตชื่อใน Firebase Auth ด้วย
136.     }
137.     if (email != null) {
138.       data['email'] = email; // เพิ่ม email ใน Map
139.     }
140.
141.     await _firestore
142.       .collection('users') // เลือก Collection 'users'
143.       .doc(user.uid) // ระบุเอกสารด้วย UID ของผู้ใช้
144.       .set(
145.         data,
146.         SetOptions(merge: true),
147.       ); // บันทึกข้อมูล (Merge: true คืออัปเดตเฉพาะ field ที่กำหนด)
148.   } catch (e) {
149.     // ตักจับข้อผิดพลาด
150.     print('Error updating user profile: $e'); // พิมพ์ข้อผิดพลาด
151.     rethrow; // โยน Exception ต่อไป
152.   }
153. }
154.
155. // Delete user settings (ลบเอกสาร Settings ของผู้ใช้)
156. static Future<void> deleteUserSettings() async {
157.   final user = _auth.currentUser; // ดึงผู้ใช้ปัจจุบัน
158.   if (user == null) return; // ถ้าไม่มีผู้ใช้ ไม่ต้องทำอะไร

```

```

159.
160.     try {
161.         await _firestore
162.             .collection('settings') // เลือก Collection 'settings'
163.                 .doc(user.uid) // ระบุเอกสารด้วย UID ของผู้ใช้
164.                     .delete(); // ลบเอกสาร
165.     } catch (e) {
166.         // ติดจับข้อผิดพลาด
167.         print('Error deleting user settings: $e'); // พิมพ์ข้อผิดพลาด
168.         rethrow; // โยน Exception ต่อไป
169.     }
170. }
171. }
```

4.2.6 services.dart

```

1. // Export all services for easy importing
2. export 'todo_service.dart';
3. export 'event_service.dart';
4. export 'auth_service.dart';
5. export 'notification_service.dart';
6. export 'settings_service.dart';
```

4.3 กลุ่ม Pages (หน้าจอหลัก)

4.3.1 begin_page.dart

```

1. import 'package:flutter/material.dart';
2. import 'package:term_project/Widgets/button_custom.dart'; // Widget ปุ่มที่กำหนดเอง
3. import 'package:term_project/Widgets/gradient_background.dart'; // Widget พื้นหลังแบบไล่เฉดสี
4. import 'package:term_project/Page/register_page.dart'; // หน้าสำหรับลงทะเบียน
5. import 'package:term_project/Page/login_page.dart'; // หน้าสำหรับเข้าสู่ระบบ
6.
7. // หน้าเริ่มต้น (Begin Page) ของแอปพลิเคชัน เป็น StatelessWidget (เมื่อจากไม่มีสถานะที่จะเปลี่ยนแปลง)
8. class BeginPage extends StatelessWidget {
9.     const BeginPage({super.key});
10.
11.    @override
12.    Widget build(BuildContext context) {
13.        return Scaffold(
14.            body: SafeArea(
15.                child: GradientBackground(
16.                    colors: const [
17.                        Color.fromRGBO(
18.                            189,
```

```
19.          187,
20.          188,
21.          245,
22.        ),
23.        Colors.white,
24.      ],
25.      stops: const [
26.        0.0,
27.        0.5,
28.      ],
29.      child: LayoutBuilder(
30.        builder: (context, constraints) {
31.          double minHeight = constraints.maxHeight;
32.
33.          // ทำให้เนื้อหาสามารถ Scroll ได้เมื่อเกิด Overflows
34.          return SingleChildScrollView(
35.            child: ConstrainedBox(
36.              constraints: BoxConstraints(
37.                minHeight: minHeight,
38.              ),
39.
40.              child: Padding(
41.                padding: const EdgeInsets.symmetric(horizontal: 40),
42.                child: Column(
43.                  mainAxisAlignment: MainAxisAlignment
44.                      .center,
45.                  crossAxisAlignment: CrossAxisAlignment
46.                      .stretch,
47.                  children: [
48.                    // Logo ของแอปพลิเคชัน
49.                    Image.asset(
50.                      'assets/logo.png',
51.                      width: 300,
52.                      height: 300,
53.                    ),
54.
55.                    const SizedBox(height: 20), // ระยะห่าง
56.                    // ข้อความต้อนรับหลัก
57.                    const Text(
58.                      'Welcome to Planity',
59.                      style: TextStyle(
60.                        fontSize: 28,
61.                        fontWeight: FontWeight.bold,
62.                        color: Color.fromRGBO(
63.                          255,
```

```
64.          80,
65.          79,
66.          120,
67.        ),
68.      ),
69.      textAlign: TextAlign.center,
70.    ),
71.
72.    const SizedBox(height: 8),
73.    // ខ្លួចរាមបន្ទាយសំណា
74.    Text(
75.      'Organize your tasks, achieve your goals',
76.      style: TextStyle(
77.        fontSize: 14,
78.        color: Colors.grey[600],
79.      ),
80.      textAlign: TextAlign.center,
81.    ),
82.
83.    const SizedBox(height: 120),
84.    // Register Button (ដែមលក្ខ)
85.    ButtonCustom(
86.      text: 'Register',
87.      onPressed: () {
88.        // នាំទაំងតីម្រោងហើយ Register
89.        Navigator.push(
90.          context,
91.          MaterialPageRoute(
92.            builder: (context) => const RegisterPage(),
93.          ),
94.        );
95.      },
96.      backgroundColor: const Color.fromARGB(
97.        255,
98.        80,
99.        79,
100.       120,
101.     ),
102.      textColor: Colors.white,
103.    ),
104.
105.    const SizedBox(height: 20), // ខ្លួចខាងខាងក្រោម
106.    // Login Button (ដែមទូទៅ)
107.    ButtonCustom(
108.      text: 'Login',
```

```

109.          onPressed: () {
110.              // นำทางไปยังหน้า Login
111.              Navigator.push(
112.                  context,
113.                  MaterialPageRoute(
114.                      builder: (context) => const LoginPage(),
115.                  ),
116.              );
117.          },
118.          backgroundColor: Colors.white,
119.          textColor: const Color.fromARGB(
120.              255,
121.              80,
122.              79,
123.              120,
124.          ),
125.          ),
126.
127.          const SizedBox(height: 40),
128.          ],
129.          ),
130.          ),
131.          ),
132.          );
133.          },
134.          ),
135.          ),
136.          ),
137.      );
138.  }
139. }
140.
141.

```

4.3.2 register_page.dart

```

1. import 'package:flutter/material.dart';
2. import 'package:firebase_auth/firebase_auth.dart'; // สำหรับการจัดการการยืนยันตัวตน (Authentication)
3. import 'package:term_project/Services/auth_service.dart'; // บริการจัดการ Firebase Auth
4. import 'package:term_project/Widgets/gradient_background.dart'; // Widget พื้นหลังใส่ระดับสี
5. import 'package:term_project/Widgets/custom_text_field.dart'; // Custom TextField ที่กำหนดเอง
6. import 'package:term_project/Widgets/custom_dialog.dart'; // Custom Dialog สำหรับแสดง Loading/Error

```

```
7. import 'package:term_project/Widgets/custom_link_text.dart'; // Widget ข้อความลิงก์
8. import 'package:term_project/Widgets/page_title.dart'; // Widget หัวข้อหน้าจอ
9. import 'package:term_project/Widgets/button_custom.dart'; // Custom Button ที่กำหนดเอง
10. import 'package:term_project/Page/login_page.dart'; // หน้าเข้าสู่ระบบ (สำหรับล็อกอิน)
11. import 'package:term_project/Page/welcome_page.dart'; // หน้าต้อนรับ/ตั้งชื่อ (สำหรับผู้ใช้ใหม่หลังลงทะเบียน)
12.
13. // คลาส RegisterPage เป็น StatefulWidget เพื่อจัดการจัดการสถานะของฟอร์ม (Text Controllers)
14. class RegisterPage extends StatefulWidget {
15.   const RegisterPage({Key? key}) : super(key: key);
16.
17.   @override
18.   State<RegisterPage> createState() => _RegisterPageState();
19. }
20.
21. class _RegisterPageState extends State<RegisterPage> {
22.   // ตัวควบคุมสำหรับบันค่าจากช่องกรอกข้อมูล
23.   final _emailController = TextEditingController();
24.   final _passwordController = TextEditingController();
25.   final _confirmPasswordController = TextEditingController(); // สำหรับยืนยันรหัสผ่าน
26.
27.   // GlobalKey สำหรับอ้างอิงและจัดการสถานะของ Form (ใช้ในการ Validate)
28.   final _formKey = GlobalKey<FormState>();
29.
30.   // ฟังก์ชันหลักในการลงทะเบียนผู้ใช้
31.   Future<void> registerUser() async {
32.     // 1. ตรวจสอบความถูกต้องของฟอร์ม (Validation)
33.     if (!_formKey.currentState!.validate()) return;
34.
35.     // 2. แสดงสถานะกำลังโหลด (Loading Dialog)
36.     // ใช้ context ที่มีไว้จาก widget
37.     CustomDialog.showLoading(context);
38.     final navigator = Navigator.of(context); // เก็บ Navigator State ก่อนมีการ await
39.
40.     try {
41.       // 3. สร้างบัญชีผู้ใช้ด้วย Firebase Auth
42.       final userCredential = await AuthService.createUserWithEmailAndPassword(
43.         email: _emailController.text.trim(),
44.         password: _passwordController.text.trim(),
45.       );
46.
47.       final user = userCredential.user;
48.       if (user == null) {
49.         // กรณีที่การสร้างบัญชีสำเร็จ แต่ไม่มี User object ('ไม่เกิดขึ้น')
50.         throw Exception('User creation failed unexpectedly.');
51.     }
52.   }
53. }
```

```
52.  
53.      // 4. สร้าง Document ผู้ใช้ใน Firestore (ตามโครงสร้างข้อมูลของแอป)  
54.      await AuthService.createUserDocument(  
55.          uid: user.uid,  
56.          email: user.email!,  
57.      );  
58.  
59.      // 5. ข่อนสถานะกำลังโหลด  
60.      navigator.pop(); // ข่อน Loading Dialog  
61.  
62.      // 6. นำทางผู้ใช้ไปยังหน้า WelcomePage เพื่อตั้งชื่อโปรไฟล์  
63.      if (mounted) {  
64.          // ใช้ pushReplacement เพื่อไม่ให้ผู้ใช้กดปุ่มย้อนกลับมาที่หน้าลงทะเบียนได้  
65.          navigator.pushReplacement(  
66.              MaterialPageRoute(builder: (context) => const WelcomePage()),  
67.          );  
68.      }  
69.  } on FirebaseAuthException catch (e) {  
70.      // 7. ตักจับข้อผิดพลาดจาก Firebase Auth  
71.      navigator.pop(); // ข่อน Loading Dialog ก่อนแสดง Error  
72.  
73.      if (mounted) {  
74.          CustomDialog.showError(  
75.              context: context,  
76.              // แปลง FirebaseAuthException เป็นข้อความที่เข้าใจง่าย  
77.              message: AuthService.getAuthErrorMessage(e),  
78.          );  
79.      }  
80.  } catch (e) {  
81.      // 8. ตักจับข้อผิดพลาดอื่น ๆ (เช่น Firestore)  
82.      navigator.pop(); // ข่อน Loading Dialog ก่อนแสดง Error  
83.  
84.      if (mounted) {  
85.          CustomDialog.showError(  
86.              context: context,  
87.              message: 'Registration failed: ${e.toString()}',  
88.          );  
89.      }  
90.  }  
91. }  
92.  
93. // Clear Controller เมื่อ Widget ถูกทำลายเพื่อป้องกัน Memory Leak  
94. @override  
95. void dispose() {  
96.     _emailController.dispose();
```



```
142.             return 'Please enter your email';
143.         }
144.         if (!RegExp(r'\S+@\S+\.\S+').hasMatch(value)) {
145.             return 'Please enter a valid email';
146.         }
147.         return null;
148.     },
149. ),
150.
151.     const SizedBox(height: 20),
152.
153.     // ช่องกรอก Password
154.     CustomTextField(
155.         controller: _passwordController,
156.         labelText: 'Password',
157.         hintText: 'Enter your password',
158.         prefixIcon: Icons.lock_outline,
159.         isPassword: true, // ตั้งค่าให้ช่องรหัสผ่านและมีปุ่ม Toggle
160.         // พิ้งก์ชันตรวจสอบ Password
161.         validator: (value) {
162.             if (value == null || value.isEmpty) {
163.                 return 'Please enter your password';
164.             }
165.             if (value.length < 6) {
166.                 return 'Password must be at least 6 characters';
167.             }
168.             return null;
169.         },
170.     ),
171.
172.     const SizedBox(height: 20),
173.
174.     // ช่องกรอก Confirm Password
175.     CustomTextField(
176.         controller: _confirmPasswordController,
177.         labelText: 'Confirm Password',
178.         hintText: 'Re-enter your password',
179.         prefixIcon: Icons.lock_outline,
180.         isPassword: true,
181.         // พิ้งก์ชันตรวจสอบ Confirm Password
182.         validator: (value) {
183.             if (value == null || value.isEmpty) {
184.                 return 'Please confirm your password';
185.             }
186.             // ตรวจสอบว่ารหัสผ่านทั้งสองช่องตรงกันหรือไม่
```

```
187.          if (value != _passwordController.text) {
188.              return 'Passwords do not match';
189.          }
190.          return null;
191.      },
192.  ),
193.
194.  const SizedBox(height: 50),
195.
196. // ปุ่ม Register
197. ButtonCustom(
198.     text: 'Register',
199.     backgroundColor: const Color.fromARGB(255, 80, 79, 120),
200.     textColor: Colors.white,
201.     onPressed: registerUser, // เรียกฟังก์ชันลงทะเบียน
202. ),
203.
204. const SizedBox(height: 20),
205.
206. // ลิงค์ไปยังหน้า Log In
207. CustomLinkText(
208.     normalText: 'Already have an account? ',
209.     linkText: 'Log In',
210.     onTap: () {
211.         Navigator.push(
212.             context,
213.             MaterialPageRoute(builder: (context) => const LoginPage()),
214.         );
215.     },
216. ),
217. ],
218. ),
219. ),
220. ),
221. ),
222. ),
223. );
224. }
225. }
226.
```

4.3.3 login_page.dart

```
1. import 'package:flutter/material.dart';
2. import 'package:firebase_auth/firebase_auth.dart'; // สำหรับการจัดการการยืนยันตัวตน (Authentication)
3. import 'package:term_project/Services/auth_service.dart'; // บริการจัดการ Firebase Auth
4. import 'package:term_project/Widgets/gradient_background.dart'; // Widget พื้นหลังไล่ระดับสี
5. import 'package:term_project/Widgets/custom_text_field.dart'; // Custom TextField ที่กำหนดเอง
6. import 'package:term_project/Widgets/custom_dialog.dart'; // Custom Dialog สำหรับแสดง Loading/Error
7. import 'package:term_project/Widgets/custom_link_text.dart'; // Widget ข้อความลิงก์
8. import 'package:term_project/Widgets/page_title.dart'; // Widget หัวข้อหน้าจอ
9. import 'package:term_project/Widgets/button_custom.dart'; // Custom Button ที่กำหนดเอง
10. import 'package:term_project/Widgets/main_navigation.dart'; // หน้าหลัก (Navigation Bar)
11. import 'package:term_project/Page/welcome_page.dart'; // หน้าต้อนรับ/ล็อกชื่อ (สำหรับผู้ใช้ใหม่)
12. import 'package:term_project/Page/register_page.dart'; // หน้าลงทะเบียน
13.
14. // คลาส LoginPage เป็น StatefulWidget เพื่อจัดการสถานะของฟอร์ม
15. class LoginPage extends StatefulWidget {
16.   const LoginPage({Key? key}) : super(key: key);
17.
18.   @override
19.   State<LoginPage> createState() => _LoginPageState();
20. }
21.
22. class _LoginPageState extends State<LoginPage> {
23.   // ตัวควบคุมสำหรับค่าจากช่องกรอกข้อมูล
24.   final _emailController = TextEditingController();
25.   final _passwordController = TextEditingController();
26.   // GlobalKey สำหรับอ้างอิงและจัดการสถานะของ Form (ใช้ในการ Validate)
27.   final _formKey = GlobalKey<FormState>();
28.
29.   // ฟังก์ชันหลักในการเข้าสู่ระบบ
30.   Future<void> loginUser() async {
31.     // 1. ตรวจสอบความถูกต้องของฟอร์ม (Validation)
32.     if (!_formKey.currentState!.validate()) return;
33.
34.     // 2. แสดงสถานะกำลังโหลด (Loading Dialog)
35.     CustomDialog.showLoading(context);
36.
37.     try {
38.       // 3. ทำการเข้าสู่ระบบด้วย Firebase Auth
39.       await AuthService.signInWithEmailAndPassword(
40.         email: _emailController.text.trim(),
41.         password: _passwordController.text.trim(),
42.       );
43.
44.       // 4. ตรวจสอบสถานะผู้ใช้
```

```
45.     final user = AuthService.currentUser;
46.     if (user == null) return; // หาก Login ไม่สำเร็จ (ควรลูกดักด้วย catch แล้ว)
47.
48.     // 5. ตรวจสอบว่าผู้ใช้ตั้งชื่อโปรไฟล์ใน Firestore หรือยัง
49.     final hasName = await AuthService.hasUserName(user.uid);
50.
51.     // 6. ซ่อนสถานะกำลังโหลด
52.     CustomDialog.hideLoading(context);
53.
54.     // 7. เน้าทางผู้ใช้ไปยังหน้าต่อไป
55.     if (mounted) {
56.         if (hasName) {
57.             // มีชื่อแล้ว → ไปที่หน้าหลัก (Main Navigation)
58.             Navigator.pushReplacement(
59.                 context,
60.                 MaterialPageRoute(builder: (context) => const MainNavigation()),
61.             );
62.         } else {
63.             // ยังไม่มีชื่อ → ไปที่หน้าต้อนรับ (WelcomePage) เพื่อตั้งชื่อ
64.             Navigator.pushReplacement(
65.                 context,
66.                 MaterialPageRoute(builder: (context) => const WelcomePage()),
67.             );
68.         }
69.     }
70. } on FirebaseAuthException catch (e) {
71.     // 8. ตักจับข้อผิดพลาดจาก Firebase Auth
72.     CustomDialog.hideLoading(context);
73.
74.     if (mounted) {
75.         CustomDialog.showError(
76.             context: context,
77.             // แปลง FirebaseAuthException เป็นข้อความที่เข้าใจง่าย
78.             message: AuthService.getAuthErrorMessage(e),
79.         );
80.     }
81. } catch (e) {
82.     // 9. ตักจับข้อผิดพลาดอื่น ๆ ที่ไม่ได้มาจากการ Firebase Auth
83.     CustomDialog.hideLoading(context);
84.
85.     if (mounted) {
86.         CustomDialog.showError(
87.             context: context,
88.             message: 'Login failed: ${e.toString()}',
89.         );

```

```
90.      }
91.    }
92.  }
93.
94. // Clear Controller ដើម្បី Widget តុកទៅលាយ
95. @override
96. void dispose() {
97.   _emailController.dispose();
98.   _passwordController.dispose();
99.   super.dispose();
100. }
101.
102. @override
103. Widget build(BuildContext context) {
104.   return Scaffold(
105.     resizeToAvoidBottomInset: true,
106.     appBar: AppBar(
107.       title: const Text(''),
108.       backgroundColor: Colors.white,
109.       elevation: 0,
110.       leading: IconButton(
111.         icon: const Icon(
112.           Icons.arrow_back,
113.           color: Color.fromARGB(255, 80, 79, 120),
114.         ),
115.         onPressed: () => Navigator.pop(context),
116.       ),
117.     ),
118.     // ឯ្យ GradientBackground បើជាបន្ទាន់សង្គម Body
119.     body: GradientBackground(
120.       child: SafeArea(
121.         child: SingleChildScrollView(
122.           padding: const EdgeInsets.symmetric(horizontal: 80, vertical: 100),
123.           child: Form(
124.             key: _formKey, // ផ្នែក Form តួអូក GlobalKey
125.             child: Column(
126.               crossAxisAlignment: CrossAxisAlignment.stretch,
127.               children: [
128.                 // ចំណាំ 'Log In'
129.                 const PageTitle(title: 'Log In'),
130.                 const SizedBox(height: 30),
131.
132.                 // ចំណាំ Email
133.                 CustomTextField(
134.                   controller: _emailController,
```

```
135.          labelText: 'Email',
136.          hintText: 'Enter your email',
137.          prefixIcon: Icons.email_outlined,
138.          keyboardType: TextInputType.emailAddress,
139.          // ພຶກຂັ້ນດຽວຈຳອົບ Email
140.          validator: (value) {
141.            if (value == null || value.isEmpty) {
142.              return 'Please enter your email';
143.            }
144.            if (!RegExp(r'\S+@\S+\.\S+').hasMatch(value)) {
145.              return 'Please enter a valid email';
146.            }
147.            return null;
148.          },
149.        ),
150.
151.        const SizedBox(height: 20),
152.
153.        // ຂອງກຮອກ Password
154.        CustomTextField(
155.          controller: _passwordController,
156.          labelText: 'Password',
157.          hintText: 'Enter your password',
158.          prefixIcon: Icons.lock_outline,
159.          isPassword: true, // ດັ່ງຄໍາໄຫ້ຂອນຮັສຜ່ານແລະມີປຸນ Toggle
160.          // ພຶກຂັ້ນດຽວຈຳອົບ Password
161.          validator: (value) {
162.            if (value == null || value.isEmpty) {
163.              return 'Please enter your password';
164.            }
165.            return null;
166.          },
167.        ),
168.
169.        const SizedBox(height: 40),
170.
171.        // ໃນ Login
172.        ButtonCustom(
173.          text: 'Log In',
174.          onPressed: loginUser, // ເຊິ່ງກຳທັນເຂົ້າສູ່ຮູບບນ
175.          backgroundColor: const Color.fromRGBO(255, 80, 79, 120),
176.          textColor: Colors.white,
177.        ),
178.
179.        const SizedBox(height: 20),
```

```

180.
181.          // ສິນກໍໄປຢັງໜ້າລົງທະເມີນ
182.          CustomLinkText(
183.              normalText: "Don't have an account? ",
184.              linkText: 'Register',
185.              onTap: () {
186.                  Navigator.push(
187.                      context,
188.                      MaterialPageRoute(
189.                          builder: (context) => const RegisterPage(),
190.                      ),
191.                  );
192.              },
193.              ),
194.              ],
195.              ),
196.              ),
197.              ),
198.              ),
199.          ),
200.      );
201.  }
202. }
```

4.3.4 welcome_page.dart

```

1.  import 'package:flutter/material.dart';
2.  import 'package:term_project/Widgets/button_custom.dart';
3.  import 'package:firebase_auth/firebase_auth.dart';
4.  import 'package:cloud_firestore/cloud_firestore.dart';
5.  import 'package:term_project/Page/home_page.dart';
6.  import 'package:term_project/Widgets/main_navigation.dart';
7.  import 'package:term_project/Widgets/gradient_background.dart';
8.  // ໜ້າຕອນຮັບຜູ້ໃຫມ່ຫລັງຈາກລົງທະເມີນ
9.  class WelcomePage extends StatefulWidget {
10.    const WelcomePage({Key? key}) : super(key: key);
11.
12.    @override
13.    State<WelcomePage> createState() => _WelcomePageState();
14.  }
15.
16.  class _WelcomePageState extends State<WelcomePage> {
17.    final _nameController = TextEditingController();
18.    final _formKey = GlobalKey<FormState>();
```

```
19.
20.     Future<void> _startApp() async {
21.       if (!_formKey.currentState!.validate()) return;
22.
23.       showDialog(
24.         context: context,
25.         barrierDismissible: false,
26.         builder: (context) => const Center(child: CircularProgressIndicator()),
27.       );
28.
29.       final navigator = Navigator.of(context);
30.
31.       try {
32.         final user = FirebaseAuth.instance.currentUser;
33.
34.         if (user != null) {
35.           // เพิ่มชื่อลงใน Document ที่สร้างไว้
36.           await FirebaseFirestore.instance
37.             .collection('users')
38.             .doc(user.uid)
39.             .update({ // ใช้ update() ไม่ได้ set()
40.               'name': _nameController.text.trim(),
41.             });
42.
43.         navigator.pop(); // นำ CircularProgressIndicator
44.
45.         // ไปหน้า HomePage
46.         Navigator.pushReplacement(
47.           context,
48.           MaterialPageRoute(builder: (context) => const MainNavigation()),
49.         );
50.       }
51.     } catch (e) {
52.       navigator.pop();
53.       showDialog(
54.         context: context,
55.         builder: (context) => AlertDialog(
56.           title: const Text('Error'),
57.           content: Text('Failed to save name: $e'),
58.           actions: [
59.             TextButton(
60.               onPressed: () => Navigator.pop(context),
61.               child: const Text('OK'),
62.             ),
63.           ],

```

```
64.      ),
65.    );
66.  }
67. }
68.
69. @override
70. void dispose() {
71.   _nameController.dispose();
72.   super.dispose();
73. }
74.
75. @override
76. Widget build(BuildContext context) {
77.   return Scaffold(
78.     body: GradientBackground(
79.       child: SafeArea(
80.         child: SingleChildScrollView(
81.           padding: const EdgeInsets.symmetric(horizontal: 80, vertical: 100),
82.           child: Form(
83.             key: _formKey,
84.             child: Column(
85.               crossAxisAlignment: CrossAxisAlignment.center,
86.               children: [
87.                 const SizedBox(height: 80),
88.
89.                 // Welcome To Planity
90.                 const Text(
91.                   'Welcome',
92.                   style: TextStyle(
93.                     fontSize: 40,
94.                     fontWeight: FontWeight.bold,
95.                     color: Color.fromARGB(255, 80, 79, 120),
96.                   ),
97.                 ),
98.                 const SizedBox(height: 10),
99.                 const Text(
100.                   'To Planity',
101.                   style: TextStyle(
102.                     fontSize: 32,
103.                     fontWeight: FontWeight.w500,
104.                     color: Color.fromARGB(255, 80, 79, 120),
105.                   ),
106.                 ),
107.
108.                 const SizedBox(height: 60),
```

```
109.  
110.        // Hello! What's your name?  
111.        const Text(  
112.            "Hello! What's your name?",  
113.            style: TextStyle(  
114.                fontSize: 20,  
115.                //fontWeight: FontWeight.w600,  
116.                color: Colors.black,  
117.            ),  
118.        ),  
119.  
120.        const SizedBox(height: 20),  
121.  
122.        // အသာဆုံး  
123.  
124.        TextFormField(  
125.            controller: _nameController,  
126.            keyboardType: TextInputType.emailAddress,  
127.            decoration: InputDecoration(  
128.                labelText: 'Your Name',  
129.                hintText: 'Enter your name',  
130.                border: OutlineInputBorder(  
131.                    borderRadius: BorderRadius.circular(12),  
132.                ),  
133.                contentPadding: const EdgeInsets.symmetric(vertical: 30, horizontal: 16),  
134.            ),  
135.            validator: (value) {  
136.                if (value == null || value.isEmpty) {  
137.                    return 'Please enter your name';  
138.                }  
139.                return null;  
140.            },  
141.        ),  
142.  
143.        const SizedBox(height: 50),  
144.  
145.        // အသာဆုံး Let's Start  
146.        ButtonCustom(  
147.            text: "Let's Start",  
148.            backgroundColor: const Color.fromARGB(255, 80, 79, 120),  
149.            textColor: Colors.white,  
150.            onPressed: _startApp,  
151.        ),  
152.    ],  
153.),
```

```
154.        ),
155.        ),
156.        ),
157.        ),
158.    );
159. }
160. }
161.
```

4.3.5 home_page.dart

```
1. import 'package:flutter/material.dart';
2. import 'package:firebase_auth/firebase_auth.dart';
3. import 'package:cloud_firestore/cloud_firestore.dart';
4. import 'dart:convert';
5. import 'package:http/http.dart' as http;
6. import 'package:intl/intl.dart';
7. import 'package:term_project/Page/begin_page.dart';
8. import 'dart:async';
9. import 'package:term_project/Model/todo_model.dart';
10. import 'package:term_project/Model/event_model.dart';
11. import 'package:term_project/Services/todo_service.dart';
12. import 'package:term_project/Services/event_service.dart';
13. import 'package:term_project/Widgets/event_card.dart';
14. import 'package:term_project/Widgets/todo_card.dart';
15. import 'package:term_project/Widgets/event_dialog.dart';
16. import 'package:term_project/Widgets/todo_dialog.dart';
17. import 'package:term_project/Widgets/group_dialog.dart';
18. import 'package:flutter_speed_dial/flutter_speed_dial.dart';
19.
20. class HomePage extends StatefulWidget {
21.   const HomePage({Key? key}) : super(key: key);
22.
23.   @override
24.   State<HomePage> createState() => _HomePageState();
25. }
26.
27. class _HomePageState extends State<HomePage> {
28.   String userName = '';
29.   String quote = 'Loading...';
30.   String author = '';
31.   bool isLoadingQuote = true;
32.   String currentDate = '';
33.   int _currentCarouselIndex = 0;
34.
```

```

35. // Event variables
36. List<EventModel> events = [];
37. bool isLoadingEvents = true;
38. String? _lastAddedEventId;
39.
40. // To-do variables
41. List<TodoModel> todos = [];
42. List<TodoGroup> groups = [];
43. String? selectedGroupId;
44. bool isLoadingTodos = true;
45. bool showCompleted = true;
46.
47. final PageController _pageController = PageController(viewportFraction: 0.6);
48.
49. StreamSubscription<List<EventModel>>? _eventsSubscription;
50. StreamSubscription<List<TodoModel>>? _todosSubscription;
51. StreamSubscription<List<TodoGroup>>? _groupsSubscription;
52. StreamSubscription<DocumentSnapshot>? _userSubscription;
53.
54. // เริ่มต้นสถานะ: เรียกใช้ฟังก์ชันที่เริ่มการฟังข้อมูลแบบเรียลไทม์
55. @override
56. void initState() {
57.   super.initState();
58.   _listenToUserName();
59.   _fetchQuote();
60.   _loadCurrentDate();
61.   _listenToEvents();
62.   _listenToGroups();
63.   _listenToTodos();
64. }
65.
66. // ท่าลายสถานะ: ยกเลิกการสมัครสมาชิก (Cancel StreamSubscription) ทั้งหมดเพื่อป้องกัน Memory Leak
67. @override
68. void dispose() {
69.   _eventsSubscription?.cancel();
70.   _pageController.dispose();
71.   _todosSubscription?.cancel();
72.   _groupsSubscription?.cancel();
73.   _userSubscription?.cancel();
74.   super.dispose();
75. }
76.
77. // ฟังการเปลี่ยนแปลงชื่อผู้ใช้แบบเรียลไทม์จาก Firestore โดยใช้ snapshots()
78. void _listenToUserName() {
79.   final user = FirebaseAuth.instance.currentUser;

```

```

80.      if (user == null) return;
81.
82.      _userSubscription = FirebaseFirestore.instance
83.          .collection('users')
84.          .doc(user.uid)
85.          .snapshots()
86.          .listen((doc) {
87.            if (doc.exists && mounted) {
88.              setState(() {
89.                userName = doc.data()?[ 'name' ] ?? 'User';
90.              });
91.              print('☑ Username updated to: $userName');
92.            }
93.          });
94.      }
95.      // ฟังการเปลี่ยนแปลงรายการ Event แบบเรียลไทม์, จัดเรียงตามวันครบกำหนด (deadline), และมีกลไก Auto-scroll ไปยัง Event ที่เพิ่งเพิ่มใหม่ (_lastAddedEventId)
96.      void _listenToEvents() {
97.        _eventsSubscription = EventService.getEventsStream().listen((eventList) {
98.          print('⚠ Real-time update: Found ${eventList.length} events');
99.
100.         setState(() {
101.             events = eventList..sort((a, b) => a.deadline.compareTo(b.deadline));
102.             isLoadingEvents = false;
103.         });
104.
105.         // Auto-scroll to newly added event
106.         if (_lastAddedEventId != null) {
107.             final targetIndex = events.indexWhere((event) => event.id == _lastAddedEventId);
108.             if (targetIndex >= 0 && events.isNotEmpty) {
109.                 WidgetsBinding.instance.addPostFrameCallback((_) {
110.                     Future.delayed(const Duration(milliseconds: 500), () {
111.                         if (mounted && _pageController.hasClients) {
112.                             _pageController.animateToPage(
113.                               targetIndex,
114.                               duration: const Duration(milliseconds: 600),
115.                               curve: Curves.easeInOut,
116.                           );
117.                         setState(() {
118.                             _currentCarouselIndex = targetIndex;
119.                         });
120.                         print('☑ Animated to event at index: $targetIndex');
121.
122.                         Future.delayed(const Duration(milliseconds: 800), () {
123.                             if (mounted) {

```

```
124.             _lastAddedEventId = null;
125.         }
126.     });
127. }
128. });
129. });
130. }
131. }
132. });
133. }
134.
135. // ดึงค่าคอมประจวันจาก API ภายนอก (https://zenquotes.io/api/today) และอัปเดต quote/author
136. Future<void> _fetchQuote() async {
137.     try {
138.         var response = await http.get(
139.             Uri.parse('https://corsproxy.io/?https://zenquotes.io/api/today'),
140.         );
141.
142.         if (response.statusCode == 200) {
143.             var data = jsonDecode(response.body);
144.             setState(() {
145.                 quote = data[0]['q'];
146.                 author = data[0]['a'];
147.                 isLoadingQuote = false;
148.             });
149.         } else {
150.             setState(() {
151.                 quote = 'Could not load quote';
152.                 isLoadingQuote = false;
153.             });
154.         }
155.     } catch (e) {
156.         setState(() {
157.             quote = 'Error loading quote';
158.             isLoadingQuote = false;
159.         });
160.     }
161. }
162.
163. // จัดรูปแบบและตั้งค่าวันที่ปัจจุบันลงในตัวแปร currentDate
164. void _loadCurrentDate() {
165.     final now = DateTime.now();
166.     final formatter = DateFormat('EEEE, d MMMM');
167.     setState(() {
168.         currentDate = formatter.format(now);
```

```

169.     });
170.   }
171.
172. // ===== To-do Functions =====
173. // ฟังการเปลี่ยนแปลงรายการ To-do Group แบบเรียลไทม์, จัดเรียงตามลำดับ (order)
174. void _listenToGroups() {
175.   _groupsSubscription = TodoService.getGroupsStream().listen((groupList) {
176.     setState(() {
177.       groups = groupList..sort((a, b) => a.order.compareTo(b.order));
178.     });
179.   });
180. }
181. // ฟังการเปลี่ยนแปลงรายการ To-do แบบเรียลไทม์, จัดเรียงตามลำดับ (order)
182. void _listenToTodos() {
183.   _todosSubscription = TodoService.getTodosStream().listen((todoList) {
184.     setState(() {
185.       todos = todoList..sort((a, b) => b.order.compareTo(a.order));
186.       isLoadingTodos = false;
187.     });
188.   });
189. }
190. // ส่งคืนรายการ To-do ที่ผ่านการกรองตาม Group ที่เลือก (selectedGroupId) และสถานะเสร็จสิ้น (showCompleted)
191. List<TodoModel> _getFilteredTodos() {
192.   var filtered = todos;
193.
194.   if (selectedGroupId != null) {
195.     filtered = filtered
196.       .where((todo) => todo.groupId == selectedGroupId)
197.       .toList();
198.   }
199.
200.   if (!showCompleted) {
201.     filtered = filtered.where((todo) => !todo.isCompleted).toList();
202.   }
203.
204.   return filtered;
205. }
206.
207. // สับสถานะเสร็จสิ้น (isCompleted) ของ To-do ที่ระบุโดยเรียกใช้ TodoService.toggleTodoComplete
208. Future<void> _toggleTodoComplete(TodoModel todo) async {
209.   await TodoService.toggleTodoComplete(todo.id, todo.isCompleted);
210. }
211. // แสดง AlertDialog เพื่อยืนยันการลบ To-do และเรียกใช้ TodoService.deleteTodo พัฒนาและ SnackBar ผลลัพธ์
212. Future<void> _deleteTodo(TodoModel todo) async {
213.   final confirmed = await showDialog<bool>(

```

```
214.     context: context,
215.     builder: (context) => AlertDialog(
216.       shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(20)),
217.       title: const Text('Delete To-do'),
218.       content: Text('Are you sure you want to delete "${todo.title}"?'),
219.       actions: [
220.         TextButton(
221.           onPressed: () => Navigator.pop(context, false),
222.           child: const Text('Cancel'),
223.         ),
224.         TextButton(
225.           onPressed: () => Navigator.pop(context, true),
226.           style: TextButton.styleFrom(backgroundColor: Colors.red),
227.           child: const Text('Delete'),
228.         ),
229.       ],
230.     ),
231.   );
232.
233.   if (confirmed == true) {
234.     try {
235.       await TodoService.deleteTodo(todo.id);
236.       if (mounted) {
237.         ScaffoldMessenger.of(context).showSnackBar(
238.           SnackBar(
239.             content: Text('To-do "${todo.title}" deleted successfully!'),
240.             style: const TextStyle(color: Colors.white),
241.             backgroundColor: const Color.fromARGB(255, 80, 79, 120),
242.           ),
243.         );
244.       }
245.     } catch (e) {
246.       print('Error deleting To-do: $e');
247.       if (mounted) {
248.         ScaffoldMessenger.of(context).showSnackBar(
249.           SnackBar(
250.             content: Text('Failed to delete To-do: $e'),
251.             backgroundColor: Colors.red,
252.           ),
253.         );
254.       }
255.     }
256.   }
257. }
```

```
259. // แสดง AlertDialog เพื่อยืนยันการลบ Event และเรียกใช้ EventService.deleteEvent พร้อมแสดง Snackbar ผลลัพธ์
260.
261. Future<void> _deleteEvent(EventModel event) async {
262.   final confirmed = await showDialog<bool>(
263.     context: context,
264.     builder: (context) => AlertDialog(
265.       shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(20)),
266.       title: const Text('Delete Event'),
267.       content: Text('Are you sure you want to delete "${event.title}"?'),
268.       actions: [
269.         TextButton(
270.           onPressed: () => Navigator.pop(context, false),
271.           child: const Text('Cancel'),
272.         ),
273.         TextButton(
274.           onPressed: () => Navigator.pop(context, true),
275.           style: TextButton.styleFrom(backgroundColor: Colors.red),
276.           child: const Text('Delete'),
277.         ),
278.       ],
279.     ),
280.   );
281.
282.   if (confirmed == true) {
283.     try {
284.       await EventService.deleteEvent(event.id);
285.       if (mounted) {
286.         ScaffoldMessenger.of(context).showSnackBar(
287.           SnackBar(
288.             content: Text('Event "${event.title}" deleted successfully!',
289.               style: const TextStyle(color: Colors.white),
290.               backgroundColor: const Color.fromARGB(255, 80, 79, 120),
291.             ),
292.           );
293.       }
294.     } catch (e) {
295.       print('Error deleting event: $e');
296.       if (mounted) {
297.         ScaffoldMessenger.of(context).showSnackBar(
298.           SnackBar(
299.             content: Text('Failed to delete event: $e'),
300.             backgroundColor: Colors.red,
301.           ),
302.         );
303.       }
304.     }
305.   }
306. }
```

```
304.     }
305.   }
306. }
307.
308. // แสดง EventDialog (ฟอร์มเพิ่ม/แก้ไข Event) และจัดการการเรียกใช้ EventService.addEvent หรือ EventService.updateEvent
309. void _showEventDialog({EventModel? event}) {
310.   showDialog(
311.     context: context,
312.     builder: (context) => EventDialog(
313.       event: event,
314.       onSave: (title, deadline, color) async {
315.         try {
316.           if (event != null) {
317.             // Update existing event
318.             await EventService.updateEvent(
319.               eventId: event.id,
320.               title: title,
321.               deadline: deadline,
322.               borderColor: color,
323.             );
324.             if (mounted) {
325.               ScaffoldMessenger.of(context).showSnackBar(
326.                 const SnackBar(
327.                   content: Text('Event updated successfully!'),
328.                   backgroundColor: Colors.green,
329.                   behavior: SnackBarBehavior.floating,
330.                   duration: Duration(seconds: 3),
331.                 ),
332.               );
333.             }
334.           } else {
335.             // Add new event
336.             final eventId = await EventService.addEvent(
337.               title: title,
338.               deadline: deadline,
339.               borderColor: color,
340.             );
341.             _lastAddedEventId = eventId;
342.             print('☑ Added event with ID: $_lastAddedEventId');
343.
344.             if (mounted) {
345.               ScaffoldMessenger.of(context).showSnackBar(
346.                 SnackBar(
347.                   content: Row(
```

```
348.          children: [
349.            const Icon(Icons.check_circle, color: Colors.white),
350.            const SizedBox(width: 12),
351.            Expanded(
352.              child: Column(
353.                mainAxisSize: MainAxisSize.min,
354.                mainAxisAlignment: MainAxisAlignment.start,
355.                children: [
356.                  const Text(
357.                    'Event Added Successfully!',
358.                    style: TextStyle(
359.                      fontWeight: FontWeight.bold,
360.                      fontSize: 16,
361.                    ),
362.                  ),
363.                  Text(
364.                    '"$title" has been added',
365.                    style: const TextStyle(fontSize: 14),
366.                  ),
367.                ],
368.              ),
369.            ),
370.          ],
371.        ),
372.        backgroundColor: Colors.green,
373.        behavior: SnackBarBehavior.floating,
374.        duration: const Duration(seconds: 3),
375.        shape: RoundedRectangleBorder(
376.          borderRadius: BorderRadius.circular(12),
377.        ),
378.        margin: const EdgeInsets.all(16),
379.      ),
380.    );
381.  }
382. }
383. } catch (e) {
384.   print('Error ${event != null ? 'updating' : 'adding'} event: $e');
385.   if (mounted) {
386.     ScaffoldMessenger.of(context).showSnackBar(
387.       SnackBar(
388.         content: Text(
389.           'Error ${event != null ? 'updating' : 'adding'} event: $e',
390.         ),
391.         backgroundColor: Colors.red,
392.         behavior: SnackBarBehavior.floating,
```

```
393.             duration: const Duration(seconds: 3),
394.         ),
395.     );
396. }
397. }
398. },
399. ),
400. );
401. }
402.
403. // แสดง TodoDialog (ฟอร์มเพิ่ม/แก้ไข To-do) และจัดการการเรียกใช้ TodoService.addTodo หรือ TodoService.updateTodo
404. void _showTodoDialog({TodoModel? todo}) {
405.   showDialog(
406.     context: context,
407.     builder: (context) => TodoDialog(
408.       todo: todo,
409.       groups: groups,
410.       onSave: (title, description, emoji, groupId) async {
411.         try {
412.           if (todo != null) {
413.             // Update existing todo
414.             await TodoService.updateTodo(
415.               todoId: todo.id,
416.               title: title,
417.               description: description,
418.               emoji: emoji,
419.               groupId: groupId,
420.             );
421.           } else {
422.             // Add new todo
423.             await TodoService.addTodo(
424.               title: title,
425.               description: description,
426.               emoji: emoji,
427.               groupId: groupId,
428.             );
429.           }
430.         } catch (e) {
431.           print('Error ${todo != null ? 'updating' : 'adding'} todo: $e');
432.           if (mounted) {
433.             ScaffoldMessenger.of(context).showSnackBar(
434.               SnackBar(
435.                 content: Text('Error ${todo != null ? 'updating' : 'adding'} todo: $e'),
436.                 backgroundColor: Colors.red,
437.               ),
438.             );
439.           }
440.         }
441.       }
442.     );
443.   }
444. }
```

```

438.           );
439.       }
440.     }
441.   },
442. ),
443. );
444. }
445. // แสดง GroupDialog (ฟอร์มเพิ่ม Group) และเรียกใช้ TodoService.addGroup
446. void _showGroupDialog() {
447.   showDialog(
448.     context: context,
449.     builder: (context) => GroupDialog(
450.       onSave: (name, emoji) async {
451.         try {
452.           await TodoService.addGroup(name: name, emoji: emoji);
453.         } catch (e) {
454.           print('Error adding group: $e');
455.           if (mounted) {
456.             ScaffoldMessenger.of(context).showSnackBar(
457.               SnackBar(
458.                 content: Text('Error adding group: $e'),
459.                 backgroundColor: Colors.red,
460.               ),
461.             );
462.           }
463.         }
464.       },
465.     ),
466.   );
467. }
468.
469. // เมธอดหลักในการสร้าง UI ทั้งหมดของหน้าจอ Scaffold , ส่วนแสดง Event, ส่วนรายการ To-do, และ Floating Action Button
470. @override
471. Widget build(BuildContext context) {
472.   return Scaffold(
473.     backgroundColor: Colors.white,
474.     body: SingleChildScrollView(
475.       child: Column(
476.         children: [
477.           const SizedBox(height: 10),
478.           SafeArea(
479.             child: Padding(
480.               padding: const EdgeInsets.symmetric(
481.                 horizontal: 20,
482.                 vertical: 10,

```

```
483.        ),
484.        child: Column(
485.          mainAxisAlignment: MainAxisAlignment.center,
486.          children: [
487.            Row(
488.              mainAxisAlignment: MainAxisAlignment.spaceBetween,
489.              crossAxisAlignment: CrossAxisAlignment.start,
490.              children: [
491.                Expanded(
492.                  flex: 2,
493.                  child: Column(
494.                    crossAxisAlignment: CrossAxisAlignment.start,
495.                    children: [
496.                      Container(
497.                        padding: const EdgeInsets.symmetric(
498.                          horizontal: 12,
499.                          vertical: 4,
500.                        ),
501.                        decoration: BoxDecoration(
502.                          color: Colors.white.withOpacity(0.8),
503.                          borderRadius: BorderRadius.circular(20),
504.                        ),
505.                        child: Row(
506.                          mainAxisSize: MainAxisSize.min,
507.                          children: [
508.                            GestureDetector(
509.                              onTap: () {
510.                                Navigator.of(context).push(
511.                                  MaterialPageRoute(
512.                                    builder: (context) => BeginPage(),
513.                                  ),
514.                                );
515.                              },
516.                              child: Container(
517.                                width: 30,
518.                                height: 30,
519.                                decoration: BoxDecoration(
520.                                  borderRadius: BorderRadius.circular(
521.                                    8,
522.                                  ),
523.                                ),
524.                                child: ClipRRect(
525.                                  borderRadius: BorderRadius.circular(
526.                                    8,
527.                                  ),
```

```
528.                child: Image.asset(
529.                    'logo.png',
530.                    fit: BoxFit.cover,
531.                ),
532.                ),
533.                ),
534.            ),
535.            const SizedBox(width: 6),
536.            Flexible(
537.                child: Text(
538.                    'Hi! $userName',
539.                    style: const TextStyle(
540.                        fontSize: 25,
541.                        fontWeight: FontWeight.w600,
542.                        color: Color.fromARGB(
543.                            255,
544.                            80,
545.                            79,
546.                            120,
547.                        ),
548.                    ),
549.                    overflow: TextOverflow.ellipsis,
550.                ),
551.            ),
552.        ],
553.    ),
554.),
555. Container(
556.    padding: const EdgeInsets.symmetric(
557.        horizontal: 12,
558.        vertical: 0,
559.    ),
560.    decoration: BoxDecoration(
561.        borderRadius: BorderRadius.circular(15),
562.    ),
563.    child: Row(
564.        mainAxisSize: MainAxisSize.min,
565.        children: [
566.            const SizedBox(width: 6),
567.            Flexible(
568.                child: Text(
569.                    currentDate,
570.                    style: const TextStyle(
571.                        fontSize: 11,
572.                        fontWeight: FontWeight.w500,
```



```
618.          children: [
619.            Text(
620.              "$quote",
621.              style: const TextStyle(
622.                fontSize: 12,
623.                fontStyle: FontStyle.italic,
624.                color: Color.fromARGB(
625.                  255,
626.                  80,
627.                  79,
628.                  120,
629.                ),
630.              ),
631.              maxLines: 2,
632.              overflow: TextOverflow.ellipsis,
633.            ),
634.            if (author.isNotEmpty) ...[
635.              const SizedBox(height: 4),
636.              Text(
637.                '- $author',
638.                style: TextStyle(
639.                  fontSize: 10,
640.                  color: Colors.grey[600],
641.                ),
642.              ),
643.            ],
644.          ],
645.        ),
646.      ),
647.    ),
648.  ],
649.),
650.],
651.),
652.),
653.),
654.
655.  const SizedBox(height: 30),
656.
657.  // Events Section
658.  isLoadingEvents
659.    ? const Center(
660.      child: Padding(
661.        padding: EdgeInsets.all(40.0),
662.        child: CircularProgressIndicator(),
```

```
663.          ),
664.          )
665.        : events.isEmpty
666.        ? Center(
667.          child: Padding(
668.            padding: const EdgeInsets.all(40.0),
669.            child: Column(
670.              children: [
671.                Icon(
672.                  Icons.event_note,
673.                  size: 64,
674.                  color: Colors.grey[400],
675.                ),
676.                const SizedBox(height: 16),
677.                Text(
678.                  'No events found',
679.                  style: TextStyle(
680.                    fontSize: 18,
681.                    color: Colors.grey[600],
682.                    fontWeight: FontWeight.w500,
683.                  ),
684.                ),
685.                const SizedBox(height: 8),
686.                Text(
687.                  'Add your first event to get started!',
688.                  style: TextStyle(
689.                    fontSize: 14,
690.                    color: Colors.grey[500],
691.                  ),
692.                ),
693.              ],
694.            ),
695.          ),
696.        )
697.      : SizedBox(
698.        height: 200,
699.        child: PageView.builder(
700.          controller: _pageController,
701.          itemCount: events.length,
702.          onPageChanged: (index) {
703.            setState(() {
704.              _currentCarouselIndex = index;
705.            });
706.          },
707.          itemBuilder: (context, index) {
```

```
708.         final event = events[index];
709.         final isActive = index == _currentCarouselIndex;
710.
711.         return EventCard(
712.             event: event,
713.             isActive: isActive,
714.             onEdit: (event) => _showEventDialog(event: event),
715.             onDelete: (event) => _deleteEvent(event),
716.         );
717.     },
718. ),
719. ),
720.
721. const SizedBox(height: 20),
722.
723. if (!isLoadingEvents && events.isNotEmpty)
724. Row(
725.     mainAxisAlignment: MainAxisAlignment.center,
726.     children: events.asMap().entries.map((entry) {
727.         return AnimatedContainer(
728.             duration: const Duration(milliseconds: 300),
729.             width: _currentCarouselIndex == entry.key ? 24 : 8,
730.             height: 8,
731.             margin: const EdgeInsets.symmetric(horizontal: 4),
732.             decoration: BoxDecoration(
733.                 borderRadius: BorderRadius.circular(4),
734.                 color: _currentCarouselIndex == entry.key
735.                     ? const Color.fromRGBO(255, 80, 79, 120)
736.                     : Colors.grey[300],
737.             ),
738.         );
739.     }).toList(),
740. ),
741.
742. const SizedBox(height: 30),
743.
744. // TO-DO LIST SECTION
745. Container(
746.     padding: const EdgeInsets.symmetric(horizontal: 40),
747.     child: Column(
748.         crossAxisAlignment: CrossAxisAlignment.start,
749.         children: [
750.             // Header
751.             Row(
752.                 mainAxisAlignment: MainAxisAlignment.spaceBetween,
```

```
753.           children: [
754.             const Text(
755.               'My To-do List',
756.               style: TextStyle(
757.                 fontSize: 24,
758.                 fontWeight: FontWeight.bold,
759.                 color: Color.fromARGB(255, 80, 79, 120),
760.               ),
761.             ),
762.             Row(
763.               children: [
764.                 // Show/Hide Completed
765.                 IconButton(
766.                   icon: Icon(
767.                     showCompleted
768.                     ? Icons.visibility
769.                     : Icons.visibility_off,
770.                     color: const Color.fromARGB(255, 80, 79, 120),
771.                   ),
772.                   onPressed: () {
773.                     setState(() {
774.                       showCompleted = !showCompleted;
775.                     });
776.                   },
777.                   tooltip: showCompleted
778.                     ? 'Hide completed'
779.                     : 'Show completed',
780.                   ),
781.
782.                 // Group Dropdown with current selection
783.                 Container(
784.                   padding: const EdgeInsets.symmetric(horizontal: 4),
785.                   decoration: BoxDecoration(
786.                     border: Border.all(
787.                       color: const Color.fromARGB(
788.                         255,
789.                         80,
790.                         79,
791.                         120,
792.                       ).withOpacity(0.3),
793.                     ),
794.                     borderRadius: BorderRadius.circular(8),
795.                   ),
796.                   child: PopupMenuButton<String>(
797.                     child: Padding(
```



```
888.         const Spacer(),
889.         const Icon(
890.             Icons.check,
891.             color: Color.fromARGB(
892.                 255,
893.                 80,
894.                 79,
895.                 120,
896.             ),
897.             size: 20,
898.         ),
899.     ],
900. ],
901. ),
902. ),
903. ),
904.
905. // Add Group
906. const PopupMenuDivider(),
907. PopupMenuItem<String>(
908.     value: 'add_group',
909.     child: Row(
910.         children: [
911.             const Icon(
912.                 Icons.add_circle_outline,
913.                 color: Color.fromARGB(255, 80, 79, 120),
914.                 size: 20,
915.             ),
916.             const SizedBox(width: 12),
917.             const Text(
918.                 'Add Group',
919.                 style: TextStyle(
920.                     color: Color.fromARGB(
921.                         255,
922.                         80,
923.                         79,
924.                         120,
925.                     ),
926.                     fontWeight: FontWeight.w600,
927.                 ),
928.             ),
929.         ],
930.     ),
931. ),
932. ],
```

```
933.          onSelected: (value) {
934.              if (value == 'add_group') {
935.                  _showGroupDialog();
936.              } else if (value == 'all') {
937.                  setState(() {
938.                      selectedGroupId =
939.                          null;
940.                  });
941.              } else {
942.                  setState(() {
943.                      selectedGroupId = value;
944.                  });
945.              }
946.          },
947.      ),
948.      ),
949.      ],
950.      ),
951.      ],
952.      ),
953.
954.      const SizedBox(height: 16),
955.
956.      if (groups.isNotEmpty) const SizedBox(height: 20),
957.
958.      // To-do List
959.
960.      Container(
961.          height: 300,
962.          decoration: BoxDecoration(
963.              borderRadius: BorderRadius.circular(12),
964.          ),
965.          child: isLoadingTodos
966.              ? const Center(
967.                  child: Padding(
968.                      padding: EdgeInsets.all(40.0),
969.                      child: CircularProgressIndicator(),
970.                  ),
971.              )
972.              : _getFilteredTodos().isEmpty
973.                  ? _buildEmptyTodoState()
974.                  : ReorderableListView.builder(
975.                      itemCount: _getFilteredTodos().length,
976.                      onReorder: _onTodoReorder,
977.                      itemBuilder: (context, index) {
```

```
978.         final todo = _getFilteredTodos()[index];
979.         return TodoCard(
980.             key: ValueKey(todo.id),
981.             todo: todo,
982.             onToggleComplete: () => _toggleTodoComplete(todo),
983.             onEdit: (todo) => _showTodoDialog(todo: todo),
984.             onDelete: (todo) => _deleteTodo(todo),
985.         );
986.     },
987. ),
988. ),
989.
990. ],
991. ),
992. ),
993. ],
994.
995. ),
996.
997. ),
998.
999. floatingActionButton: SpeedDial(
1000.     icon: Icons.add,
1001.     activeIcon: Icons.close,
1002.     backgroundColor: const Color.fromARGB(255, 80, 79, 120),
1003.     foregroundColor: Colors.white,
1004.     overlayColor: Colors.black,
1005.     overlayOpacity: 0.4,
1006.     spacing: 12,
1007.     children: [
1008.         SpeedDialChild(
1009.             child: const Icon(Icons.check_circle_outline),
1010.             label: 'Add To-do',
1011.             backgroundColor: const Color.fromARGB(255, 80, 79, 120),
1012.             foregroundColor: Colors.white,
1013.             onTap: () => _showTodoDialog(),
1014.         ),
1015.         SpeedDialChild(
1016.             child: const Icon(Icons.event),
1017.             label: 'Add Event',
1018.             backgroundColor: const Color.fromARGB(255, 80, 79, 120),
1019.             foregroundColor: Colors.white,
1020.             onTap: () => _showEventDialog(),
1021.         ),
1022.     ],

```

```
1023.            ),
1024.        );
1025.    }
1026.
1027.    Widget _buildEmptyTodoState() {
1028.        return Center(
1029.            child: Padding(
1030.                padding: const EdgeInsets.all(40.0),
1031.                child: Column(
1032.                    children: [
1033.                        Icon(Icons.check_circle_outline, size: 80, color: Colors.grey[300]),
1034.                        const SizedBox(height: 16),
1035.                        Text(
1036.                            selectedGroupId == null
1037.                                ? 'No to-dos yet'
1038.                                : 'No to-dos in this group',
1039.                            style: TextStyle(
1040.                                fontSize: 18,
1041.                                color: Colors.grey[600],
1042.                                fontWeight: FontWeight.w500,
1043.                            ),
1044.                        ),
1045.                        const SizedBox(height: 8),
1046.                        Text(
1047.                            'Tap the + button to add one',
1048.                            style: TextStyle(fontSize: 14, color: Colors.grey[500]),
1049.                        ),
1050.                    ],
1051.                ),
1052.            ),
1053.        );
1054.    }
1055.
1056.    void _onTodoReorder(int oldIndex, int newIndex) async {
1057.        if (oldIndex < newIndex) {
1058.            newIndex -= 1;
1059.        }
1060.
1061.        final filteredTodos = _getFilteredTodos();
1062.        final reorderedList = List<TodoModel>.from(filteredTodos);
1063.
1064.        // Reorder ໃບ list ຂໍາຄරາ
1065.        final movedTodo = reorderedList.removeAt(oldIndex);
1066.        reorderedList.insert(newIndex, movedTodo);
1067.
```

```

1068.        try {
1069.            await TodoService.reorderTodos(reorderedList);
1070.        } catch (e) {
1071.            print('Error reordering todos: $e');
1072.            // แสดง snackbar ถ้า error
1073.            if (mounted) {
1074.                ScaffoldMessenger.of(context).showSnackBar(
1075.                    const SnackBar(content: Text('Failed to reorder todos')),
1076.                );
1077.            }
1078.        }
1079.    }
1080.}

```

4.3.6 notification_page.dart

```

1. import 'package:flutter/material.dart';
2. import 'package:intl/intl.dart';
3. import 'dart:async';
4. import 'package:term_project/Services/notification_service.dart';
5.
6. //รายการการแจ้งเตือนที่ถึงมา, แต่ละรายการเป็น Map ที่เก็บชื่อ默เล่น title, message, deadline, icon, color
7. class NotificationPage extends StatefulWidget {
8.   const NotificationPage({super.key});
9.
10.  @override
11.  State<NotificationPage> createState() => _NotificationPageState();
12. }
13.
14. class _NotificationPageState extends State<NotificationPage> {
15.   List<Map<String, dynamic>> notifications = [];
16.   bool isLoading = true;
17.
18.   StreamSubscription<List<Map<String, dynamic>>>? _eventsSubscription;
19.
20.   @override
21.   void initState() {
22.     super.initState();
23.     _listenToNotifications();
24.   }
25.
26.   @override
27.   void dispose() {
28.     _eventsSubscription?.cancel();
29.     super.dispose();

```

```
30.    }
31.
32. //เริ่มการฟัง (Subscribe) Stream ของการแจ้งเตือนจาก NotificationService.getNotificationsStream(). เมื่อข้อมูลอัปเดตจะทำการ
     setState เพื่อแสดงรายการการแจ้งเตือนใหม่ และตั้งค่า isLoading = false
33. void _listenToNotifications() {
34.   _eventsSubscription = NotificationService.getNotificationsStream().listen(
35.     (notificationList) {
36.       if (mounted) {
37.         setState(() {
38.           notifications = notificationList;
39.           isLoading = false;
40.         });
41.       }
42.       print('☑ Notifications updated: ${notificationList.length} items');
43.     },
44.     onError: (error) {
45.       print('☒ Error listening to notifications: $error');
46.       if (mounted) {
47.         setState(() => isLoading = false);
48.       }
49.     },
50.   );
51. }
52.
53. @override
54. Widget build(BuildContext context) {
55.   return Scaffold(
56.     backgroundColor: Colors.white,
57.     appBar: AppBar(
58.       backgroundColor: Colors.white,
59.       foregroundColor: const Color.fromARGB(255, 80, 79, 120),
60.       titleSpacing: 20,
61.       title: Padding(
62.         padding: const EdgeInsets.only(top: 20),
63.         child: Text(
64.           'Notifications',
65.           style: const TextStyle(
66.             fontSize: 25,
67.             fontWeight: FontWeight.w600,
68.             color: Color.fromARGB(255, 80, 79, 120),
69.           ),
70.           overflow: TextOverflow.ellipsis,
71.         ),
72.       ),
73.       elevation: 0,
```

```
74.      ),
75.      body: isLoading
76.          ? const Center(child: CircularProgressIndicator())
77.          : notifications.isEmpty
78.          ? _buildEmptyState()
79.          : ListView.builder(
80.              padding: const EdgeInsets.all(16),
81.              itemCount: notifications.length,
82.              itemBuilder: (context, index) {
83.                  return _buildNotificationCard(notifications[index]);
84.              },
85.          ),
86.      ),
87.  );
88. }
89. //Widget ที่แสดงเมื่อ notifications ว่างเปล่า โดยแสดงไอคอนและข้อความแจ้งว่าไม่มีการแจ้งเตือน
90. Widget _buildEmptyState() {
91.     return Center(
92.         child: Column(
93.             mainAxisAlignment: MainAxisAlignment.center,
94.             children: [
95.                 Icon(Icons.notifications_none, size: 100, color: Colors.grey[300]),
96.                 const SizedBox(height: 16),
97.                 Text(
98.                     'No Notifications',
99.                     style: TextStyle(
100.                         fontSize: 20,
101.                         fontWeight: FontWeight.bold,
102.                         color: Colors.grey[600],
103.                     ),
104.                 ),
105.                 const SizedBox(height: 8),
106.                 Text(
107.                     'You have no upcoming event deadlines',
108.                     style: TextStyle(fontSize: 14, color: Colors.grey[500]),
109.                 ),
110.             ],
111.         ),
112.     );
113. }
114.
115. // Widget ที่สร้างการ์ด (Card) สำหรับการแจ้งเตือนแต่ละรายการ
116. Widget _buildNotificationCard(Map<String, dynamic> notification) {
117.     final deadline = notification['deadline'] as DateTime;
118.     final dateFormat = DateFormat('d MMM yyyy, HH:mm');
```

```
119.
120.    return Card(
121.      margin: const EdgeInsets.only(bottom: 12),
122.      elevation: 2,
123.      shape: RoundedRectangleBorder(
124.        borderRadius: BorderRadius.circular(16),
125.        side: BorderSide(color: notification['borderColor'], width: 2),
126.      ),
127.      child: InkWell(
128.        onTap: () {
129.        },
130.        borderRadius: BorderRadius.circular(16),
131.        child: Padding(
132.          padding: const EdgeInsets.all(16),
133.          child: Row(
134.            children: [
135.              // Icon
136.              Container(
137.                width: 50,
138.                height: 50,
139.                decoration: BoxDecoration(
140.                  color: notification['iconColor'].withOpacity(0.1),
141.                  borderRadius: BorderRadius.circular(12),
142.                ),
143.                child: Icon(
144.                  notification['icon'],
145.                  color: notification['iconColor'],
146.                  size: 28,
147.                ),
148.              ),
149.              const SizedBox(width: 16),
150.
151.              // Content
152.              Expanded(
153.                child: Column(
154.                  crossAxisAlignment: CrossAxisAlignment.start,
155.                  children: [
156.                    // Title
157.                    Text(
158.                      notification['title'],
159.                      style: const TextStyle(
160.                        fontSize: 16,
161.                        fontWeight: FontWeight.bold,
162.                        color: Color.fromRGBO(255, 80, 79, 120),
163.                      ),
164.                  ],
165.                ),
166.              ),
167.            ],
168.          ),
169.        ),
170.      ),
171.    );
172.
```

```
164.          maxLines: 2,
165.          overflow: TextOverflow.ellipsis,
166.        ),
167.        const SizedBox(height: 4),
168.
169.        // Message
170.        Text(
171.          notification['message'],
172.          style: TextStyle(
173.            fontSize: 14,
174.            color: notification['iconColor'],
175.            fontWeight: FontWeight.w600,
176.          ),
177.        ),
178.        const SizedBox(height: 4),
179.
180.        // Deadline
181.        Row(
182.          children: [
183.            Icon(
184.              Icons.access_time,
185.              size: 14,
186.              color: Colors.grey[600],
187.            ),
188.            const SizedBox(width: 4),
189.            Text(
190.              dateFormat.format(deadline),
191.              style: TextStyle(
192.                fontSize: 12,
193.                color: Colors.grey[600],
194.              ),
195.            ),
196.          ],
197.        ),
198.      ],
199.    ),
200.  ),
201.
202.  // Badge
203.  Container(
204.    padding: const EdgeInsets.symmetric(
205.      horizontal: 12,
206.      vertical: 6,
207.    ),
208.    decoration: BoxDecoration(
```

```
209.          color: notification['iconColor'].withOpacity(0.15),
210.          borderRadius: BorderRadius.circular(20),
211.        ),
212.        child: Text(
213.          notification['type'] == 'today' ? 'TODAY' : 'TOMORROW',
214.          style: TextStyle(
215.            fontSize: 12,
216.            fontWeight: FontWeight.bold,
217.            color: notification['iconColor'],
218.          ),
219.        ),
220.      ),
221.    ],
222.  ),
223. ),
224. ),
225. );
226. }
227. }
228.
```

4.3.7 setting_page.dart

```
1. import 'package:flutter/material.dart';
2. import 'package:term_project/Services/auth_service.dart';
3. import 'package:term_project/Services/settings_service.dart';
4. import 'package:term_project/Page/begin_page.dart';
5. import 'package:term_project/Page/login_page.dart';
6.
7. class SettingsPage extends StatefulWidget {
8.   const SettingsPage({super.key});
9.
10.  @override
11.  State<SettingsPage> createState() => _SettingsPageState();
12. }
13.
14. class _SettingsPageState extends State<SettingsPage> {
15.   final primaryColor = const Color.fromARGB(255, 80, 79, 120);
16.
17.   bool _notificationsEnabled = true;
18.   bool _soundEnabled = true;
19.   String _userName = '';
20.   String _userEmail = '';
21.
22.   @override
```

```
23. void initState() {
24.   super.initState();
25.   _loadUserData();
26.   _loadSettings();
27. }
28.
29. // ดึงชื่อ และ อีเมล ของผู้ใช้มาจาก AuthService และ SettingsService
30. Future<void> _loadUserData() async {
31.   final user = AuthService.currentUser;
32.   if (user == null) return;
33.
34.   try {
35.     final userProfile = await SettingsService.getUserProfile();
36.     setState(() {
37.       _userName = userProfile['name'] ?? 'User';
38.       _userEmail = user.email ?? 'No Email';
39.     });
40.   } catch (e) {
41.     print('Error loading user data: $e');
42.   }
43. }
44.
45. // ดึงการตั้งค่าผู้ใช้จาก SettingsService และอัปเดตสถานะของสวิทช์ในหน้า Settings
46. Future<void> _loadSettings() async {
47.   try {
48.     final settings = await SettingsService.getUserSettings();
49.     setState(() {
50.       _notificationsEnabled = settings['notifications'] ?? true;
51.       _soundEnabled = settings['sound'] ?? true;
52.     });
53.   } catch (e) {
54.     print('Error loading settings: $e');
55.   }
56. }
57.
58. // บันทึกการตั้งค่าผู้ใช้โดยเรียกใช้ฟังก์ชัน saveUserSettings ของ SettingsService
59. Future<void> _saveSettings() async {
60.   try {
61.     await SettingsService.saveUserSettings(
62.       notifications: _notificationsEnabled,
63.       sound: _soundEnabled,
64.     );
65.
66.     if (mounted) {
67.       ScaffoldMessenger.of(context).showSnackBar(
```

```
68.         const SnackBar(
69.             content: Text('Settings saved!'),
70.             duration: Duration(seconds: 1),
71.             behavior: SnackBarBehavior.floating,
72.         ),
73.     );
74. }
75. } catch (e) {
76.     print('Error saving settings: $e');
77. }
78. }
79.
80. // แสดงกล่องโต๊ะคอมเพื่อแก้ไขชื่อผู้ใช้
81. Future<void> _showEditNameDialog() async {
82.     final controller = TextEditingController(text: _userName);
83.
84.     await showDialog(
85.         context: context,
86.         builder: (context) => AlertDialog(
87.             shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(20)),
88.             title: const Text('Edit Name'),
89.             content: TextField(
90.                 controller: controller,
91.                 decoration: const InputDecoration(
92.                     labelText: 'Display Name',
93.                     border: OutlineInputBorder(),
94.                 ),
95.                 maxLength: 30,
96.             ),
97.             actions: [
98.                 TextButton(
99.                     onPressed: () => Navigator.pop(context),
100.                     child: const Text('Cancel'),
101.                 ),
102.                 ElevatedButton(
103.                     onPressed: () async {
104.                         final newName = controller.text.trim();
105.                         if (newName.isNotEmpty) {
106.                             Navigator.pop(context); // ✅ ปิด dialog ก่อน
107.
108.                             setState(() => _userName = newName);
109.
110.                         // บันทึกใน Firebase
111.                         try {
112.                             await SettingsService.updateUserProfile(name: newName);

```

```
113.
114.            if (mounted) {
115.                ScaffoldMessenger.of(context).showSnackBar(
116.                    const SnackBar(
117.                        content: Text('Name updated!'),
118.                        duration: Duration(seconds: 2),
119.                        behavior: SnackBarBehavior.floating,
120.                    ),
121.                );
122.            }
123.        } catch (e) {
124.            if (mounted) {
125.                ScaffoldMessenger.of(context).showSnackBar(
126.                    SnackBar(
127.                        content: Text('Failed to update name: $e'),
128.                        behavior: SnackBarBehavior.floating,
129.                    ),
130.                );
131.            }
132.        }
133.    },
134.    style: ElevatedButton.styleFrom(
135.        backgroundColor: primaryColor,
136.        foregroundColor: Colors.white,
137.    ),
138.    child: const Text('Save'),
139. ),
140. ],
141. ),
142. ),
143. );
144. }
145.
146. // แสดงกล่องโต๊ะตอบยืนยันการลบบัญชีผู้ใช้
147. Future<void> _showDeleteAccountDialog() async {
148.     final bool confirm =
149.         await showDialog(
150.             context: context,
151.             builder: (context) => AlertDialog(
152.                 shape: RoundedRectangleBorder(
153.                     borderRadius: BorderRadius.circular(20),
154.                 ),
155.                 title: const Row(
156.                     children: [
157.                         Icon(Icons.warning, color: Colors.red),
```

```
158.           SizedBox(width: 8),
159.           Text('Delete Account'),
160.       ],
161.   ),
162.   content: const Text(
163.     'Are you sure you want to delete your account? This action cannot be undone and all your data will
be permanently deleted.',
164.     style: TextStyle(fontSize: 14),
165.   ),
166.   actions: [
167.     TextButton(
168.       onPressed: () =>
169.         Navigator.pop(context, false), // ส่ง false เมื่อยกเลิก
170.         child: const Text('Cancel'),
171.     ),
172.     ElevatedButton(
173.       onPressed: () =>
174.         Navigator.pop(context, true), // ส่ง true เมื่อยืนยัน
175.         style: ElevatedButton.styleFrom(
176.           backgroundColor: Colors.red,
177.           foregroundColor: Colors.white,
178.         ),
179.         child: const Text('Delete'),
180.     ),
181.   ],
182. ),
183. ) ??
184. false;
185.
186. // ถ้าผู้ใช้ไม่ยืนยัน (กด Cancel หรือ ปิด Dialog) ก็หยุดการทำงาน
187. if (!confirm) {
188.   return;
189. }
190.
191. // --- ส่วนที่เพิ่ม Loading (Full-Screen) ---
192.
193. // 1. แสดง Loading Dialog (CircularProgressIndicator) แบบเต็มหน้าจอ
194. //    ใช้ context เดิมของ SettingsPage
195. showDialog(
196.   context: context,
197.   barrierDismissible: false, // บล็อกหน้าจอ
198.   builder: (context) =>
199.     const Center(child: CircularProgressIndicator(color: Colors.red)),
200. );
201.
```

```
202.     try {
203.         // 2. กระบวนการลบ
204.         await AuthService.deleteUserData(AuthService.currentUser!.uid);
205.         await AuthService.deleteUser();
206.
207.         if (mounted) {
208.             // 3. นำทางไปที่ BeginPage และล้างหน้าทั้งหมดออกจาก Stack
209.             // ค่าสั่งนี้จะแทนที่ Loading Dialog ที่กำลังแสดงอยู่โดยอัตโนมัติ
210.             Navigator.of(context).pushAndRemoveUntil(
211.                 MaterialPageRoute(builder: (context) => const BeginPage()),
212.                 (Route<dynamic> route) => false,
213.             );
214.
215.             ScaffoldMessenger.of(context).showSnackBar(
216.                 const SnackBar(
217.                     content: Text('Account deleted successfully!'),
218.                     backgroundColor: Colors.green,
219.                     duration: Duration(seconds: 3),
220.                 ),
221.             );
222.         }
223.     } catch (e) {
224.         if (mounted) {
225.             // 4. หากมี Error: ต้องปิด Loading Dialog ก่อนแสดง Error Snackbar
226.             // ใช้ pop() เพื่อปิด Dialog ที่มี CircularProgressIndicator
227.             Navigator.of(context).pop();
228.
229.             ScaffoldMessenger.of(context).showSnackBar(
230.                 SnackBar(
231.                     content: Text('Error: ${e.toString()}'),
232.                     backgroundColor: Colors.red,
233.                 ),
234.             );
235.         }
236.     }
237. }
238.
239. // แสดงกล่องโต๊ดตอบยืนยันการออกจากระบบ
240. Future<void> _logout() async {
241.     await showDialog(
242.         context: context,
243.         builder: (context) => AlertDialog(
244.             shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(20)),
245.             title: const Text('Logout'),
246.             content: const Text('Are you sure you want to logout?'),
```

```
247.     actions: [
248.       TextButton(
249.         onPressed: () => Navigator.pop(context),
250.         child: const Text('Cancel'),
251.       ),
252.       ElevatedButton(
253.         onPressed: () async {
254.           await AuthService.signOut();
255.           if (mounted) {
256.             Navigator.of(context).pushReplacement(
257.               MaterialPageRoute(builder: (context) => const LoginPage()),
258.             );
259.           }
260.         },
261.         style: ElevatedButton.styleFrom(
262.           backgroundColor: primaryColor,
263.           foregroundColor: Colors.white,
264.         ),
265.         child: const Text('Logout'),
266.       ),
267.     ],
268.   ),
269. );
270. }
271.
272. @override
273. Widget build(BuildContext context) {
274.   return Scaffold(
275.     backgroundColor: Colors.grey[50],
276.     appBar: AppBar(
277.       backgroundColor: Colors.white,
278.       foregroundColor: const Color.fromARGB(255, 80, 79, 120),
279.       titleSpacing: 20, // ⌂ ເວັນຮະຍະຈາກຂອນໜ້າຍ
280.       title: Padding(
281.         padding: const EdgeInsets.only(top: 20),
282.         child: Text(
283.           'Settings',
284.           style: const TextStyle(
285.             fontSize: 25,
286.             fontWeight: FontWeight.w600,
287.             color: Color.fromARGB(255, 80, 79, 120),
288.           ),
289.           overflow: TextOverflow.ellipsis,
290.         ),
291.       ),

```

```
292.           elevation: 0,
293.         ),
294.
295.       body: SingleChildScrollView(
296.         child: Column(
297.           children: [
298.             SizedBox(height: 20),
299.             Container(
300.               margin: const EdgeInsets.symmetric(horizontal: 16),
301.               width: double.infinity,
302.               padding: const EdgeInsets.all(24),
303.               decoration: BoxDecoration(
304.                 gradient: LinearGradient(
305.                   colors: [primaryColor, primaryColor.withOpacity(0.7)],
306.                   begin: Alignment.bottomCenter,
307.                   end: Alignment.topCenter,
308.                 ),
309.                 borderRadius: BorderRadius.circular(24),
310.               ),
311.               child: Column(
312.                 children: [
313.                   // Name
314.                   Text(
315.                     _userName,
316.                     style: const TextStyle(
317.                       fontSize: 28,
318.                       fontWeight: FontWeight.bold,
319.                       color: Colors.white,
320.                     ),
321.                   ),
322.                   const SizedBox(height: 8),
323.
324.                   // Email
325.                   Text(
326.                     _userEmail,
327.                     style: TextStyle(
328.                       fontSize: 16,
329.                       color: Colors.white.withOpacity(0.9),
330.                     ),
331.                   ),
332.                   const SizedBox(height: 20),
333.
334.                   // Edit Profile Button
335.                   OutlinedButton.icon(
336.                     onPressed: _showEditNameDialog,
```

```
337.          icon: const Icon(Icons.edit, size: 18),
338.          label: const Text('Edit Name'),
339.          style: OutlinedButton.styleFrom(
340.            foregroundColor: Colors.white,
341.            side: const BorderSide(color: Colors.white, width: 2),
342.            shape: RoundedRectangleBorder(
343.              borderRadius: BorderRadius.circular(20),
344.            ),
345.            padding: const EdgeInsets.symmetric(
346.              horizontal: 24,
347.              vertical: 12,
348.            ),
349.          ),
350.        ),
351.      ],
352.    ),
353.  ),
354.
355.  const SizedBox(height: 16),
356.
357. // Settings Sections
358. _buildSection(
359.   title: 'Preferences',
360.   children: [
361.     _buildSwitchTile(
362.       icon: Icons.notifications,
363.       title: 'Notifications',
364.       subtitle: 'Receive reminders for your tasks',
365.       value: _notificationsEnabled,
366.       onChanged: (value) {
367.         setState(() => _notificationsEnabled = value);
368.         _saveSettings();
369.       },
370.     ),
371.   ],
372. ),
373.
374. _buildSection(
375.   title: 'About',
376.   children: [
377.     _buildTile(
378.       icon: Icons.info,
379.       title: 'About App',
380.       subtitle: 'Version 1.0.0',
381.       onTap: () => _showAboutDialog(),
```

```
382.          ),
383.          ],
384.        ),
385.
386.        _buildSection(
387.          title: 'Account',
388.          children: [
389.            _buildTile(
390.              icon: Icons.logout,
391.              title: 'Logout',
392.              subtitle: 'Sign out of your account',
393.              onTap: _logout,
394.              iconColor: Colors.orange,
395.            ),
396.            _buildTile(
397.              icon: Icons.delete_forever,
398.              title: 'Delete Account',
399.              subtitle: 'Permanently delete your account',
400.              onTap: _showDeleteAccountDialog,
401.              iconColor: Colors.red,
402.            ),
403.          ],
404.        ),
405.
406.        const SizedBox(height: 32),
407.      ],
408.    ),
409.  ),
410. );
411. }
412.
413. // ส่วนส่วน (Section) ในหน้า Settings โดยมีหัวข้อและรายการย่อย
414. Widget _buildSection({
415.   required String title,
416.   required List<Widget> children,
417. }) {
418.   return Column(
419.     mainAxisAlignment: MainAxisAlignment.start,
420.     children: [
421.       Padding(
422.         padding: const EdgeInsets.fromLTRB(16, 16, 16, 8),
423.         child: Text(
424.           title,
425.           style: TextStyle(
426.             fontSize: 14,
```

```
427.           fontWeight: FontWeight.bold,
428.           color: Colors.grey[600],
429.         ),
430.       ),
431.     ),
432.     Container(
433.       margin: const EdgeInsets.symmetric(horizontal: 16),
434.       decoration: BoxDecoration(
435.         color: Colors.white,
436.         borderRadius: BorderRadius.circular(12),
437.         boxShadow: [
438.           BoxShadow(
439.             color: Colors.black.withOpacity(0.05),
440.             blurRadius: 10,
441.             offset: const Offset(0, 2),
442.           ),
443.         ],
444.       ),
445.       child: Column(children: children),
446.     ),
447.   ],
448. );
449. }
450.
451. // ส่วนรายการแบบ ListTile สำหรับการตั้งค่าท้าไป
452. Widget _buildTile({
453.   required IconData icon,
454.   required String title,
455.   required String subtitle,
456.   required VoidCallback onTap,
457.   Color? iconColor,
458. }) {
459.   return ListTile(
460.     leading: Container(
461.       padding: const EdgeInsets.all(8),
462.       decoration: BoxDecoration(
463.         color: (iconColor ?? primaryColor).withOpacity(0.1),
464.         borderRadius: BorderRadius.circular(8),
465.       ),
466.       child: Icon(icon, color: iconColor ?? primaryColor, size: 24),
467.     ),
468.     title: Text(title, style: const TextStyle(fontWeight: FontWeight.w600)),
469.     subtitle: Text(
470.       subtitle,
471.       style: TextStyle(color: Colors.grey[600], fontSize: 12),
```

```
472.      ),
473.      trailing: const Icon(Icons chevron_right, color: Colors.grey),
474.      onTap: onTap,
475.    );
476.  }
477.
478. // สร้างรายการแบบ SwitchListTile สำหรับการตั้งค่าที่เปิด/ปิดได้
479. Widget _buildSwitchTile({
480.   required IconData icon,
481.   required String title,
482.   required String subtitle,
483.   required bool value,
484.   required ValueChanged<bool> onChanged,
485. }) {
486.   return SwitchListTile(
487.     secondary: Container(
488.       padding: const EdgeInsets.all(8),
489.       decoration: BoxDecoration(
490.         color: primaryColor.withOpacity(0.1),
491.         borderRadius: BorderRadius.circular(8),
492.       ),
493.       child: Icon(icon, color: primaryColor, size: 24),
494.     ),
495.     title: Text(title, style: const TextStyle(fontWeight: FontWeight.w600)),
496.     subtitle: Text(
497.       subtitle,
498.       style: TextStyle(color: Colors.grey[600], fontSize: 12),
499.     ),
500.     value: value,
501.     activeColor: primaryColor,
502.     onChanged: onChanged,
503.   );
504. }
505.
506. // แสดงกล่องโต้ตอบข้อมูลเกี่ยวกับแอป
507. void _showAboutDialog() {
508.   showDialog(
509.     context: context,
510.     builder: (context) => AlertDialog(
511.       shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(20)),
512.       title: const Text('About Planity'),
513.       content: const Column(
514.         mainAxisAlignment: MainAxisAlignment.min,
515.         crossAxisAlignment: CrossAxisAlignment.start,
516.         children: [
```

```
517.           Text('Version: 1.0.0', style: TextStyle(fontSize: 14)),
518.           SizedBox(height: 8),
519.           Text(
520.             'A simple and powerful task management app.',
521.             style: TextStyle(fontSize: 14),
522.           ),
523.           SizedBox(height: 16),
524.           Text(
525.             'Developed with ❤ using Flutter',
526.             style: TextStyle(fontSize: 12, color: Colors.grey),
527.           ),
528.         ],
529.       ),
530.       actions: [
531.         TextButton(
532.           onPressed: () => Navigator.pop(context),
533.           child: const Text('Close'),
534.         ),
535.       ],
536.     ),
537.   );
538. }
539. }
```

4.3.8 pages.dart

```
1. // Export all pages for easy importing
2. export 'begin_page.dart';
3. export 'home_page.dart';
4. export 'login_page.dart';
5. export 'notification_page.dart';
6. export 'register_page.dart';
7. export 'setting_page.dart';
8. export 'welcome_page.dart';
```

4.4 กู้ม Widgets (ส่วนประกอบ UI ที่ใช้ซ้ำ)

4.4.1 button_custom.dart

```
1. import 'package:flutter/material.dart';
2.
3. class ButtonCustom extends StatelessWidget {
4.   final String text;
5.   final VoidCallback onPressed; // คือฟังก์ชันที่ถูกเรียกเมื่อปุ่มถูกกด
6.   final Color backgroundColor; // เปลี่ยนชื่อจาก color เพื่อให้ชัดเจนขึ้นว่าเป็นสีพื้นหลัง
7.   final Color textColor; // เพิ่มพารามิเตอร์สำหรับสีข้อความ
8.
9.   const ButtonCustom({
10.     super.key,
11.     required this.text,
12.     required this.onPressed,
13.     required this.backgroundColor, // ต้องกำหนดเมื่อเรียกใช้
14.     required this.textColor, // ต้องกำหนดเมื่อเรียกใช้
15.   });
16.
17. @override
18. Widget build(BuildContext context) {
19.   return Container(
20.     width: 340,
21.     height: 50,
22.     decoration: BoxDecoration(
23.       borderRadius: BorderRadius.circular(8),
24.       boxShadow: [
25.         BoxShadow(
26.           // ใช้สีพื้นหลังที่รับเข้ามา หรือสีเทาเข้มเป็นเงา
27.           color: backgroundColor.withOpacity(0.3),
28.           blurRadius: 8,
29.           offset: const Offset(0, 4),
30.         ),
31.       ],
32.     ),
33.     child: ElevatedButton(
34.       onPressed: onPressed,
35.       style: ElevatedButton.styleFrom(
36.         backgroundColor: backgroundColor, // **ใช้ค่า backgroundColor ที่รับเข้ามา**
37.         elevation: 0, // ปิด shadow ของปุ่ม (ไข้ของ Container แทน)
38.         shape: RoundedRectangleBorder(
39.           borderRadius: BorderRadius.circular(16),
40.         ),
41.       ),
42.       child: Text(
```

```

43.         text,
44.         style: TextStyle(
45.             fontSize: 18,
46.             fontWeight: FontWeight.bold,
47.             color: textColor, // **ໃນນີ້ຄ່າ textColor ທີ່ຮັບເຂົ້າມາ**
48.         ),
49.     ),
50. ),
51. );
52. }
53. }
```

4.4.2 custom_text_field.dart

```

5 import 'package:flutter/material.dart';
6
7 class CustomTextField extends StatefulWidget {
8   final TextEditingController controller;
9   final String labelText;
10  final String hintText;
11  final IconData prefixIcon;
12  final bool isPassword;
13  final TextInputType? keyboardType;
14  final String? Function(String?)? validator;
15
16  const CustomTextField({
17    Key? key,
18    required this.controller,
19    required this.labelText,
20    required this.hintText,
21    required this.prefixIcon,
22    this.isPassword = false,
23    this.keyboardType,
24    this.validator,
25  }) : super(key: key);
26
27  @override
28  State<CustomTextField> createState() => _CustomTextFieldState();
29 }
30
31 class _CustomTextFieldState extends State<CustomTextField> {
32   bool _obscureText = true;
33
34   @override
35   Widget build(BuildContext context) {
36     return TextFormField(
37       controller: widget.controller,
```

```
38      obscureText: widget.isPassword ? _obscureText : false,
39      keyboardType: widget.keyboardType,
40      decoration: InputDecoration(
41        labelText: widget.labelText,
42        hintText: widget.hintText,
43        prefixIcon: Icon(widget.prefixIcon),
44        border: OutlineInputBorder(
45          borderRadius: BorderRadius.circular(12),
46        ),
47        suffixIcon: widget.isPassword
48        ? IconButton(
49          icon: Icon(
50            _obscureText ? Icons.visibility_off : Icons.visibility,
51          ),
52          onPressed: () {
53            setState(() {
54              _obscureText = !_obscureText;
55            });
56          },
57        )
58        : null,
59      ),
60      validator: widget.validator,
61    );
62  }
63 }
```

4.4.3 custom_dialog.dart

```
1. import 'package:flutter/material.dart';
2. import 'button_custom.dart';
3.
4. class CustomDialog {
5.   // Error Dialog
6.   static void showError({
7.     required BuildContext context,
8.     required String message,
9.     VoidCallback? onOk,
10.   }) {
11.   showDialog(
12.     context: context,
13.     builder: (context) => AlertDialog(
14.       shape: RoundedRectangleBorder(
15.         borderRadius: BorderRadius.circular(20),
16.       ),
```

```
17.         contentPadding: const EdgeInsets.all(20),
18.         content: SizedBox(
19.             width: 300,
20.             height: 150,
21.             child: Column(
22.                 mainAxisAlignment: MainAxisAlignment.center,
23.                 children: [
24.                     const Icon(
25.                         Icons.error_outline,
26.                         color: Colors.redAccent,
27.                         size: 50,
28. ),
29.                     const SizedBox(height: 15),
30.                     Text(
31.                         message,
32.                         textAlign: TextAlign.center,
33.                         style: const TextStyle(fontSize: 16),
34. ),
35.                 ],
36.             ),
37.         ),
38.         actionsAlignment: MainAxisAlignment.center,
39.         actions: [
40.             ButtonCustom(
41.                 text: 'OK',
42.                 onPressed: () {
43.                     Navigator.pop(context);
44.                     if (onOk != null) onOk();
45.                 },
46.                 backgroundColor: const Color.fromARGB(255, 80, 79, 120),
47.                 textColor: Colors.white,
48.             ),
49.         ],
50.     ),
51. );
52. }
53.
54. // Success Dialog
55. static void showSuccess({
56.     required BuildContext context,
57.     required String message,
58.     VoidCallback? onOk,
59. }) {
60.     showDialog(
61.         context: context,
```

```
62.      builder: (context) => AlertDialog(
63.        shape: RoundedRectangleBorder(
64.          borderRadius: BorderRadius.circular(20),
65.        ),
66.        contentPadding: const EdgeInsets.all(20),
67.        content: SizedBox(
68.          width: 300,
69.          height: 150,
70.          child: Column(
71.            mainAxisAlignment: MainAxisAlignment.center,
72.            children: [
73.              const Icon(
74.                Icons.check_circle_outline,
75.                color: Colors.green,
76.                size: 50,
77.              ),
78.              const SizedBox(height: 15),
79.              Text(
80.                message,
81.                textAlign: TextAlign.center,
82.                style: const TextStyle(fontSize: 16),
83.              ),
84.            ],
85.          ),
86.        ),
87.        actionsAlignment: MainAxisAlignment.center,
88.        actions: [
89.          ButtonCustom(
90.            text: 'OK',
91.            onPressed: () {
92.              Navigator.pop(context);
93.              if (onOk != null) onOk();
94.            },
95.            backgroundColor: const Color.fromRGBO(255, 80, 79, 120),
96.            textColor: Colors.white,
97.          ),
98.        ],
99.      ),
100.    );
101.  }
102.
103. // Loading Dialog
104. static void showLoading(BuildContext context) {
105.   showDialog(
106.     context: context,
```

```
107.        barrierDismissible: false,
108.        builder: (context) => const Center(
109.            child: CircularProgressIndicator(
110.                color: Color.fromARGB(255, 80, 79, 120),
111.            ),
112.        ),
113.    );
114. }
115.
116. // 隱藏 Loading
117. static void hideLoading(BuildContext context) {
118.     Navigator.of(context).pop();
119. }
120. }
121.
```

4.4.4 custom_link_text.dart

```
1. import 'package:flutter/gestures.dart';
2. import 'package:flutter/material.dart';
3.
4. class CustomLinkText extends StatelessWidget {
5.     final String normalText;
6.     final String linkText;
7.     final VoidCallback onTap;
8.
9.     const CustomLinkText({
10.         Key? key,
11.         required this.normalText,
12.         required this.linkText,
13.         required this.onTap,
14.     }) : super(key: key);
15.
16.     @override
17.     Widget build(BuildContext context) {
18.         return Center(
19.             child: RichText(
20.                 text: TextSpan(
21.                     style: TextStyle(color: Colors.grey[700], fontSize: 14),
22.                     children: [
23.                         TextSpan(text: normalText),
24.                         TextSpan(
25.                             text: linkText,
26.                             style: const TextStyle(
27.                                 color: Color.fromARGB(255, 80, 79, 120),
```

```
28.           fontWeight: FontWeight.bold,
29.           decoration: TextDecoration.underline,
30.         ),
31.         recognizer: TapGestureRecognizer()..onTap = onTap,
32.       ),
33.     ],
34.   ),
35. ),
36. );
37. }
38. }
39.
```

4.4.5 gradient_background.dart

```
1. import 'package:flutter/material.dart';
2.
3. class GradientBackground extends StatelessWidget {
4.   final Widget child;
5.   final List<Color>? colors;
6.   final List<double>? stops;
7.
8.   const GradientBackground({
9.     Key? key,
10.    required this.child,
11.    this.colors,
12.    this.stops,
13.  }) : super(key: key);
14.
15. @override
16. Widget build(BuildContext context) {
17.   return Container(
18.     height: MediaQuery.of(context).size.height,
19.     decoration: BoxDecoration(
20.       gradient: LinearGradient(
21.         begin: Alignment.bottomCenter,
22.         end: Alignment.topCenter,
23.         colors: colors ?? [
24.           const Color.fromARGB(200, 180, 182, 250),
25.           Colors.white,
26.         ],
27.         stops: stops ?? [0.0, 0.4],
28.       ),
29.     ),
30.     child: child,
```

```
31.      );
32.    }
33.  }
34.
```

4.4.6 page_title.dart

```
1.  import 'package:flutter/material.dart';
2.
3.  class PageTitle extends StatelessWidget {
4.    final String title;
5.    final Color? color;
6.
7.    const PageTitle({
8.      Key? key,
9.      required this.title,
10.     this.color,
11.   }) : super(key: key);
12.
13.   @override
14.   Widget build(BuildContext context) {
15.     return Text(
16.       title,
17.       style: TextStyle(
18.         fontSize: 32,
19.         fontWeight: FontWeight.bold,
20.         color: color ?? const Color.fromARGB(255, 80, 79, 120),
21.       ),
22.     );
23.   }
24. }
25.
```

4.4.7 main_navigation.dart

```
1.  import 'package:flutter/material.dart';
2.  import '../Services/notification_service.dart';
3.  import '../Page/home_page.dart';
4.  import '../Page/notification_page.dart';
5.  import '../Page/setting_page.dart';
6.
7.  class MainNavigation extends StatefulWidget {
8.    const MainNavigation({Key? key}) : super(key: key);
9.
10.   @override
```

```
11.     State<MainNavigation> createState() => _MainNavigationState();
12. }
13.
14. class _MainNavigationState extends State<MainNavigation> {
15.     int _currentIndex = 0;
16.     int _notificationCount = 0;
17.
18.     // รายการหน้าทั้งหมด
19.     final List<Widget> _pages = [
20.         const HomePage(),
21.         const NotificationPage(),
22.         const SettingsPage(),
23.     ];
24.
25.     @override
26.     void initState() {
27.         super.initState();
28.         _loadNotificationCount();
29.         _listenToEventChanges();
30.     }
31.
32.     // ดึงจำนวนการแจ้งเตือน
33.     Future<void> _loadNotificationCount() async {
34.         try {
35.             final count = await NotificationService.getNotificationCount();
36.             if (mounted) {
37.                 setState(() {
38.                     _notificationCount = count;
39.                 });
40.             }
41.         } catch (e) {
42.             print('Error loading notification count: $e');
43.         }
44.     }
45.
46.     // ฟังการเปลี่ยนแปลงของ Events แบบ Real-time
47.     void _listenToEventChanges() {
48.         NotificationService.getNotificationCountStream().listen((count) {
49.             if (mounted) {
50.                 setState(() {
51.                     _notificationCount = count;
52.                 });
53.             }
54.         });
55.     }
}
```

```
56.  
57.     @override  
58.     Widget build(BuildContext context) {  
59.         return Scaffold(  
60.             body: IndexedStack(  
61.                 index: _currentIndex,  
62.                 children: _pages,  
63.             ),  
64.             bottomNavigationBar: BottomNavigationBar(  
65.                 type: BottomNavigationBarType.fixed,  
66.                 backgroundColor: Colors.white,  
67.                 selectedItemColor: const Color.fromARGB(255, 80, 79, 120),  
68.                 unselectedItemColor: Colors.grey,  
69.                 selectedFontSize: 12,  
70.                 unselectedFontSize: 12,  
71.                 currentIndex: _currentIndex,  
72.                 onTap: (index) {  
73.                     setState(() {  
74.                         _currentIndex = index;  
75.                     });  
76.  
77.                     // รีเฟรชจำนวนการแจ้งเตือนเมื่อเปิดหน้า Notification  
78.                     if (index == 1) {  
79.                         _loadNotificationCount();  
80.                     }  
81.                 },  
82.                 items: [  
83.                     const BottomNavigationBarItem(  
84.                         icon: Icon(Icons.home),  
85.                         label: 'Home',  
86.                     ),  
87.                     BottomNavigationBarItem(  
88.                         icon: _buildNotificationIcon(),  
89.                         label: 'Notifications',  
90.                     ),  
91.                     const BottomNavigationBarItem(  
92.                         icon: Icon(Icons.settings),  
93.                         label: 'Settings',  
94.                     ),  
95.                 ],  
96.             ),  
97.         );  
98.     }  
99.  
100.    // สร้างไอคอน Notification พ่วง Badge
```

```

101. Widget _buildNotificationIcon() {
102.   return Stack(
103.     clipBehavior: Clip.none,
104.     children: [
105.       const Icon(Icons.notifications),
106.       if (_notificationCount > 0)
107.         Positioned(
108.           right: -6,
109.           top: -4,
110.           child: Container(
111.             padding: const EdgeInsets.all(4),
112.             decoration: BoxDecoration(
113.               color: Colors.red,
114.               shape: BoxShape.circle,
115.               border: Border.all(
116.                 color: Colors.white,
117.                 width: 1.5,
118.               ),
119.             ),
120.             constraints: const BoxConstraints(
121.               minWidth: 18,
122.               minHeight: 18,
123.             ),
124.             child: Text(
125.               _notificationCount > 9 ? '9+' : '${_notificationCount}',
126.               style: const TextStyle(
127.                 color: Colors.white,
128.                 fontSize: 10,
129.                 fontWeight: FontWeight.bold,
130.               ),
131.               textAlign: TextAlign.center,
132.             ),
133.           ),
134.         ),
135.       ],
136.     );
137. }
138. }
139.

```

4.4.8 todo_card.dart

```

1. import 'package:flutter/material.dart';
2. import '../Model/todo_model.dart';
3.

```

```
4.   class TodoCard extends StatelessWidget {
5.     final TodoModel todo;
6.     final VoidCallback? onToggleComplete;
7.     final Function(TodoModel)? onEdit;
8.     final Function(TodoModel)? onDelete;
9.
10.    const TodoCard({
11.      Key? key,
12.      required this.todo,
13.      this.onToggleComplete,
14.      this.onEdit,
15.      this.onDelete,
16.    }) : super(key: key);
17.
18.    @override
19.    Widget build(BuildContext context) {
20.      final primaryColor = const Color.fromARGB(255, 80, 79, 120);
21.
22.      return Card(
23.        key: key,
24.        margin: const EdgeInsets.only(bottom: 12),
25.        elevation: 2,
26.        shape: RoundedRectangleBorder(
27.          borderRadius: BorderRadius.circular(12),
28.          side: BorderSide(color: primaryColor, width: 2),
29.        ),
30.        child: ListTile(
31.          contentPadding: const EdgeInsets.symmetric(horizontal: 12, vertical: 8),
32.          leading: Row(
33.            mainAxisAlignment: MainAxisAlignment.min,
34.            children: [
35.              const SizedBox(width: 8),
36.              Text(todo.emoji, style: const TextStyle(fontSize: 28)),
37.            ],
38.          ),
39.          title: Text(
40.            todo.title,
41.            style: TextStyle(
42.              fontSize: 16,
43.              fontWeight: FontWeight.bold,
44.              decoration: todo.isCompleted ? TextDecoration.lineThrough : null,
45.              color: todo.isCompleted ? Colors.grey : Colors.black87,
46.            ),
47.          ),
48.          subtitle: todo.description.isNotEmpty
```

```
49.          ? Padding(
50.            padding: const EdgeInsets.only(top: 4),
51.            child: Text(
52.              todo.description,
53.              style: TextStyle(
54.                fontSize: 14,
55.                color: todo.isCompleted ? Colors.grey : Colors.black54,
56.                decoration: todo.isCompleted
57.                  ? TextDecoration.lineThrough
58.                  : null,
59.                ),
60.                maxLines: 2,
61.                overflow: TextOverflow.ellipsis,
62.              ),
63.            )
64.          : null,
65.        trailing: Row(
66.          mainAxisAlignment: MainAxisAlignment.end,
67.          children: [
68.            // Checkbox
69.            Checkbox(
70.              value: todo.isCompleted,
71.              activeColor: primaryColor,
72.              onChanged: (_)
73.                => onToggleComplete?.call(),
74.            ),
75.            // More Options
76.            Padding(
77.              padding: const EdgeInsets.only(right: 20.0),
78.              child: PopupMenuButton(
79.                itemBuilder: (context) => [
80.                  const PopupMenuItem(
81.                    value: 'edit',
82.                    child: Row(
83.                      children: [
84.                        Icon(Icons.edit, size: 20),
85.                        SizedBox(width: 8),
86.                        Text('Edit'),
87.                      ],
88.                    ),
89.                  ),
90.                  const PopupMenuItem(
91.                    value: 'delete',
92.                    child: Row(
93.                      children: [
```

```
94.           Icon(Icons.delete, size: 20, color: Colors.red),
95.           SizedBox(width: 8),
96.           Text('Delete', style: TextStyle(color: Colors.red)),
97.         ],
98.       ),
99.     ),
100.   ],
101.   onSelected: (value) {
102.     switch (value) {
103.       case 'edit':
104.         onEdit?.call(todo);
105.         break;
106.       case 'delete':
107.         onDelete?.call(todo);
108.         break;
109.       }
110.     },
111.   ),
112.   ),
113. ],
114. ),
115. ),
116. );
117. }
118. }
119.
120.
```

4.4.9 event_card.dart

```
1. import 'package:flutter/material.dart';
2. import '../Model/event_model.dart';
3.
4. class EventCard extends StatelessWidget {
5.   final EventModel event;
6.   final bool isActive;
7.   final VoidCallback? onTap;
8.   final Function(EventModel)? onEdit;
9.   final Function(EventModel)? onDelete;
10.
11. const EventCard({
12.   Key? key,
13.   required this.event,
14.   this.isActive = false,
15.   this.onTap,
```

```
16.      this.onEdit,
17.      this.onDelete,
18.    }) : super(key: key);
19.
20.  @override
21.  Widget build(BuildContext context) {
22.    return AnimatedContainer(
23.      duration: const Duration(milliseconds: 300),
24.      curve: Curves.easeInOut,
25.      height: isActive ? 150 : 100,
26.      margin: EdgeInsets.symmetric(
27.        horizontal: isActive ? 5 : 10,
28.        vertical: isActive ? 0 : 15,
29.      ),
30.      child: Opacity(
31.        opacity: isActive ? 1.0 : 0.6,
32.        child: Container(
33.          decoration: BoxDecoration(
34.            gradient: LinearGradient(
35.              begin: Alignment.topLeft,
36.              end: Alignment.bottomRight,
37.              colors: [
38.                event.borderColor,
39.                event.borderColor.withOpacity(0.7),
40.              ],
41.            ),
42.            borderRadius: BorderRadius.circular(20),
43.            boxShadow: [
44.              BoxShadow(
45.                color: event.borderColor.withOpacity(0.3),
46.                blurRadius: 15,
47.                offset: const Offset(0, 10),
48.              ),
49.            ],
50.          ),
51.          child: Material(
52.            color: Colors.transparent,
53.            child: Stack(
54.              children: [
55.                // Event Card Content
56.                InkWell(
57.                  onTap: onTap,
58.                  borderRadius: BorderRadius.circular(20),
59.                  child: Center(
60.                    child: Padding(
```

```
61.          padding: const EdgeInsets.all(10),
62.          child: Column(
63.            mainAxisAlignment: MainAxisAlignment.center,
64.            crossAxisAlignment: CrossAxisAlignment.center,
65.            children: [
66.              const SizedBox(height: 8),
67.              Text(
68.                event.title,
69.                style: const TextStyle(
70.                  fontSize: 18,
71.                  fontWeight: FontWeight.bold,
72.                  color: Colors.white,
73.                ),
74.                textAlign: TextAlign.center,
75.                maxLines: 2,
76.                overflow: TextOverflow.ellipsis,
77.              ),
78.              const SizedBox(height: 8),
79.              Container(
80.                padding: const EdgeInsets.symmetric(
81.                  horizontal: 8,
82.                  vertical: 4,
83.                ),
84.                decoration: BoxDecoration(
85.                  color: Colors.white.withOpacity(0.2),
86.                  borderRadius: BorderRadius.circular(20),
87.                ),
88.                child: Text(
89.                  event.formattedDeadline,
90.                  style: const TextStyle(
91.                    fontSize: 11,
92.                    color: Colors.white,
93.                  ),
94.                ),
95.              ),
96.              const SizedBox(height: 10),
97.              Text(
98.                event.daysUntilDeadline == 0
99.                  ? 'Today'
100.                  : '${event.daysUntilDeadline.abs()}',
101.                style: const TextStyle(
102.                  fontSize: 40,
103.                  fontWeight: FontWeight.bold,
104.                  color: Colors.white,
105.                  height: 1.0,
```

```
106.          ),
107.          ),
108.          const SizedBox(height: 4),
109.          Text(
110.            event.daysUntilDeadline == 0
111.              ? ''
112.              : (event.daysUntilDeadline > 0 ? 'DAYS LEFT' : 'DAYS SINCE'),
113.            style: const TextStyle(
114.              fontSize: 12,
115.              fontWeight: FontWeight.w600,
116.              letterSpacing: 1.0,
117.              color: Colors.white,
118.            ),
119.            ),
120.          ],
121.        ),
122.        ),
123.        ),
124.        ),
125.
126.        // More Options Button
127.        Positioned(
128.          top: 8,
129.          right: 8,
130.          child: PopupMenuButton<String>(
131.            icon: Container(
132.              padding: const EdgeInsets.all(4),
133.              decoration: BoxDecoration(
134.                color: Colors.white.withOpacity(0.3),
135.                borderRadius: BorderRadius.circular(8),
136.              ),
137.              child: const Icon(
138.                Icons.more_vert,
139.                color: Colors.white,
140.                size: 20,
141.              ),
142.            ),
143.            shape: RoundedRectangleBorder(
144.              borderRadius: BorderRadius.circular(12),
145.            ),
146.            itemBuilder: (context) => [
147.              const PopupMenuItem(
148.                value: 'edit',
149.                child: Row(
150.                  children: [
```

```
151.           Icon(Icons.edit, size: 20),
152.           SizedBox(width: 8),
153.           Text('Edit'),
154.         ],
155.       ),
156.     ),
157.     const PopupMenuItem(
158.       value: 'delete',
159.       child: Row(
160.         children: [
161.           Icon(
162.             Icons.delete,
163.             size: 20,
164.             color: Colors.red,
165.           ),
166.           SizedBox(width: 8),
167.           Text(
168.             'Delete',
169.             style: TextStyle(color: Colors.red),
170.           ),
171.         ],
172.       ),
173.     ),
174.   ],
175.   onSelected: (value) {
176.     if (value == 'edit' && onEdit != null) {
177.       onEdit!(event);
178.     } else if (value == 'delete' && onDelete != null) {
179.       onDelete!(event);
180.     }
181.   },
182. ),
183. ),
184. ],
185. ),
186. ),
187. ),
188. ),
189. );
190. }
191. }
192.
193.
```

4.4.10 todo_dialog.dart

```
1. import 'package:flutter/material.dart';
2. import '../Model/todo_model.dart';
3.
4. class TodoDialog extends StatefulWidget {
5.   final TodoModel? todo;
6.   final List<TodoGroup> groups;
7.   final Function(String title, String description, String emoji, String? groupId)? onSave;
8.
9.   const TodoDialog({
10.     Key? key,
11.     this.todo,
12.     required this.groups,
13.     this.onSave,
14.   }) : super(key: key);
15.
16.   @override
17.   State<TodoDialog> createState() => _TodoDialogState();
18. }
19.
20. class _TodoDialogState extends State<TodoDialog> {
21.   late TextEditingController _titleController;
22.   late TextEditingController _descController;
23.   late String _selectedEmoji;
24.   late String? _selectedGroupId;
25.   late bool _isEditing;
26.
27.   @override
28.   void initState() {
29.     super.initState();
30.     _isEditing = widget.todo != null;
31.     _titleController = TextEditingController(text: widget.todo?.title ?? '');
32.     _descController = TextEditingController(text: widget.todo?.description ?? '');
33.     _selectedEmoji = widget.todo?.emoji ?? '📝';
34.     _selectedGroupId = widget.todo?.groupId;
35.   }
36.
37.   @override
38.   void dispose() {
39.     _titleController.dispose();
40.     _descController.dispose();
41.     super.dispose();
42.   }
43.
44.   @override
```

```
45.     Widget build(BuildContext context) {
46.       return AlertDialog(
47.         shape: RoundedRectangleBorder(
48.           borderRadius: BorderRadius.circular(20),
49.         ),
50.         title: Text(_isEditing ? 'Edit To-do' : 'Add To-do'),
51.         content: SingleChildScrollView(
52.           child: Column(
53.             mainAxisSize: MainAxisSize.min,
54.             children: [
55.               // Emoji Selector
56.               GestureDetector(
57.                 onTap: () async {
58.                   final emoji = await _showEmojiPicker(_selectedEmoji);
59.                   if (emoji != null) {
60.                     setState(() => _selectedEmoji = emoji);
61.                   }
62.                 },
63.                 child: Container(
64.                   width: 80,
65.                   height: 80,
66.                   decoration: BoxDecoration(
67.                     color: const Color.fromARGB(255, 80, 79, 120).withOpacity(0.1),
68.                     borderRadius: BorderRadius.circular(20),
69.                   ),
70.                   child: Center(
71.                     child: Text(
72.                       _selectedEmoji,
73.                       style: const TextStyle(fontSize: 40),
74.                     ),
75.                   ),
76.                 ),
77.               ),
78.               const SizedBox(height: 16),
79.
80.               // Title
81.               TextField(
82.                 controller: _titleController,
83.                 decoration: const InputDecoration(
84.                   labelText: 'Title',
85.                   border: OutlineInputBorder(),
86.                   prefixIcon: Icon(Icons.title),
87.                 ),
88.                 maxLength: 50,
89.                 autofocus: true,
```

```
90.        ),
91.        const SizedBox(height: 8),
92.
93.        // Description
94.        TextField(
95.            controller: _descController,
96.            decoration: const InputDecoration(
97.                labelText: 'Description (optional)',
98.                border: OutlineInputBorder(),
99.                prefixIcon: Icon(Icons.description),
100.            ),
101.            maxLines: 3,
102.            maxLength: 200,
103.        ),
104.        const SizedBox(height: 8),
105.
106.        // Group Selector
107.        DropdownButtonFormField<String?>(
108.            value: _selectedGroupId,
109.            decoration: const InputDecoration(
110.                labelText: 'Group (optional)',
111.                border: OutlineInputBorder(),
112.                prefixIcon: Icon(Icons.folder),
113.            ),
114.            items: [
115.                const DropdownMenuItem(
116.                    value: null,
117.                    child: Text('No Group'),
118.                ),
119.                ...widget.groups.map(
120.                    (g) => DropdownMenuItem(
121.                        value: g.id,
122.                        child: Text('${g.emoji} ${g.name}'),
123.                    ),
124.                ),
125.            ],
126.            onChanged: (value) => setState(() => _selectedGroupId = value),
127.        ),
128.    ],
129.),
130.),
131. actions: [
132.     TextButton(
133.         onPressed: () => Navigator.pop(context),
134.         child: const Text('Cancel'),
```

```
135.        ),
136.        ElevatedButton(
137.          onPressed: () async {
138.            if (_titleController.text.trim().isEmpty) {
139.              ScaffoldMessenger.of(context).showSnackBar(
140.                const SnackBar(content: Text('Please enter a title')),
141.              );
142.            return;
143.          }
144.
145.          Navigator.pop(context);
146.          widget.onSave?.call(
147.            _titleController.text.trim(),
148.            _descController.text.trim(),
149.            _selectedEmoji,
150.            _selectedGroupId,
151.          );
152.        },
153.        style: ElevatedButton.styleFrom(
154.          backgroundColor: const Color.fromARGB(255, 80, 79, 120),
155.          foregroundColor: Colors.white,
156.        ),
157.        child: Text(_isEditing ? 'Update' : 'Add'),
158.      ),
159.    ],
160.  );
161. }
162.
163. Future<String?> _showEmojiPicker(String currentEmoji) async {
164.   const emojis = [
165.     '☒', '☒', '❖', '★', '⌚', '⌚', '⌚', '⌚', '⌚', '⌚',
166.     '⌚', '⌚', '⌚', '⌚', '⌚', '⌚', '⌚', '⌚', '⌚', '⌚',
167.     '⌚', '⌚', '⌚', '⌚', '⌚',
168.   ];
169.
170.   return showDialog<String>(
171.     context: context,
172.     builder: (context) => Dialog(
173.       shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(20)),
174.       child: Padding(
175.         padding: const EdgeInsets.all(20),
176.         child: Column(
177.           mainAxisSize: MainAxisSize.min,
178.           children: [
179.             const Text(
```

```
180.          'Choose Emoji',
181.          style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
182.        ),
183.        const SizedBox(height: 20),
184.        SizedBox(
185.          width: 300,
186.          height: 200,
187.          child: GridView.builder(
188.            gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
189.              crossAxisCount: 6,
190.              mainAxisSpacing: 10,
191.              crossAxisSpacing: 10,
192.            ),
193.            itemCount: emojis.length,
194.            itemBuilder: (context, index) {
195.              final emoji = emojis[index];
196.              return InkWell(
197.                onTap: () => Navigator.pop(context, emoji),
198.                child: Container(
199.                  decoration: BoxDecoration(
200.                    color: emoji == currentEmoji
201.                      ? const Color.fromARGB(255, 80, 79, 120).withOpacity(0.2)
202.                      : Colors.transparent,
203.                    borderRadius: BorderRadius.circular(10),
204.                  ),
205.                  child: Center(
206.                    child: Text(
207.                      emoji,
208.                      style: const TextStyle(fontSize: 24),
209.                    ),
210.                  ),
211.                ),
212.              );
213.            },
214.            ),
215.            ),
216.            ],
217.          ),
218.        ),
219.      ),
220.    );
221.  }
222. }
223.
224.
```

4.4.11 event_dialog.dart

```
1. import 'package:flutter/material.dart';
2. import 'package:intl/intl.dart';
3. import '../Model/event_model.dart';
4.
5. class EventDialog extends StatefulWidget {
6.   final EventModel? event;
7.   final Function(String title, DateTime deadline, Color color)? onSave;
8.
9.   const EventDialog({
10.     Key? key,
11.     this.event,
12.     this.onSave,
13.   }) : super(key: key);
14.
15.   @override
16.   State<EventDialog> createState() => _EventDialogState();
17. }
18.
19. class _EventDialogState extends State<EventDialog> {
20.   late TextEditingController _titleController;
21.   late DateTime _selectedDate;
22.   late Color _selectedColor;
23.   late bool _isEditing;
24.
25.   final List<Color> _eventColors = [
26.     const Color.fromARGB(255, 80, 79, 120),
27.     const Color(0xFFFF9AA2), // ធម្មអ៊ូន
28.     const Color(0xFFFFC3A0), // សំណុំនៅ
29.     const Color.fromARGB(255, 240, 204, 87), // មេត្តិកាសពាសទេលខោរោង
30.     const Color(0xFF9FE2BF), // ឱ្យយាទៅពាសទេល
31.     const Color(0xFFA0CED9), // ដឹងជីវិត
32.     const Color(0xFF8E7CC3), // នំនាំពាសទេល
33.     const Color(0xFF8CA6DB), // បាករុណអ៊ូន
34.   ];
35.
36.   @override
37.   void initState() {
38.     super.initState();
39.     _isEditing = widget.event != null;
40.     _titleController = TextEditingController(text: widget.event?.title ?? '');
41.     _selectedDate = widget.event?.deadline ?? DateTime.now();
42.     _selectedColor = widget.event?.borderColor ?? _eventColors[0];
```

```
43. }
44.
45. @override
46. void dispose() {
47.   _titleController.dispose();
48.   super.dispose();
49. }
50.
51. @override
52. Widget build(BuildContext context) {
53.   return AlertDialog(
54.     shape: RoundedRectangleBorder(
55.       borderRadius: BorderRadius.circular(20),
56.     ),
57.     title: Text(_isEditing ? 'Edit Event' : 'Add Event'),
58.     content: SingleChildScrollView(
59.       child: Column(
60.         mainAxisSize: MainAxisSize.min,
61.         children: [
62.           // Title
63.           TextField(
64.             controller: _titleController,
65.             decoration: const InputDecoration(
66.               labelText: 'Event Title',
67.               border: OutlineInputBorder(),
68.               prefixIcon: Icon(Icons.title),
69.             ),
70.             maxLength: 50,
71.             autofocus: true,
72.           ),
73.           const SizedBox(height: 16),
74.
75.           // Date Picker
76.           InkWell(
77.             onTap: () async {
78.               final picked = await showDatePicker(
79.                 context: context,
80.                 initialDate: _selectedDate,
81.                 firstDate: DateTime(2000),
82.                 lastDate: DateTime(2100),
83.               );
84.               if (picked != null) {
85.                 setState(() => _selectedDate = picked);
86.               }
87.             },
88.           ),
89.         ],
90.       ),
91.     ),
92.   );
93. }
```

```
88.          child: InputDecorator(
89.            decoration: const InputDecoration(
90.              labelText: 'Deadline',
91.              border: OutlineInputBorder(),
92.              prefixIcon: Icon(Icons.calendar_today),
93.            ),
94.            child: Text(
95.              DateFormat('d MMM yyyy').format(_selectedDate),
96.              style: const TextStyle(fontSize: 16),
97.            ),
98.          ),
99.        ),
100.      const SizedBox(height: 16),
101.
102.      // Color Picker
103.      const Align(
104.        alignment: Alignment.centerLeft,
105.        child: Text(
106.          'Event Color:',
107.          style: TextStyle(fontSize: 14, fontWeight: FontWeight.w500),
108.        ),
109.      ),
110.      const SizedBox(height: 8),
111.      Wrap(
112.        spacing: 8,
113.        runSpacing: 8,
114.        children: _eventColors.map((color) {
115.          final isSelected = _selectedColor == color;
116.          return InkWell(
117.            onTap: () => setState(() => _selectedColor = color),
118.            child: Container(
119.              width: 40,
120.              height: 40,
121.              decoration: BoxDecoration(
122.                color: color,
123.                shape: BoxShape.circle,
124.                border: Border.all(
125.                  color: isSelected ? Colors.black : Colors.transparent,
126.                  width: 3,
127.                ),
128.              ),
129.              child: isSelected
130.                ? const Icon(Icons.check, color: Colors.white)
131.                : null,
132.            ),

```

```
133.          );
134.      }).toList(),
135.      ),
136.    ],
137.  ),
138.  ),
139. actions: [
140.   TextButton(
141.     onPressed: () => Navigator.pop(context),
142.     child: const Text('Cancel'),
143.   ),
144.   ElevatedButton(
145.     onPressed: () async {
146.       if (_titleController.text.trim().isEmpty) {
147.         ScaffoldMessenger.of(context).showSnackBar(
148.           const SnackBar(content: Text('Please enter event title')),
149.         );
150.       }
151.     }
152.
153.     Navigator.pop(context);
154.     widget.onSave?.call(
155.       _titleController.text.trim(),
156.       _selectedDate,
157.       _selectedColor,
158.     );
159.   },
160.   style: ElevatedButton.styleFrom(
161.     backgroundColor: const Color.fromARGB(255, 80, 79, 120),
162.     foregroundColor: Colors.white,
163.   ),
164.   child: Text(_isEditing ? 'Update' : 'Add'),
165. ),
166. ],
167. );
168. }
169. }
170.
```

4.4.12 group_dialog.dart

```
1. import 'package:flutter/material.dart';
2.
3. class GroupDialog extends StatefulWidget {
4.   final Function(String name, String emoji)? onSave;
5.
6.   const GroupDialog({
7.     Key? key,
8.     this.onSave,
9.   }) : super(key: key);
10.
11. @override
12. State<GroupDialog> createState() => _GroupDialogState();
13. }
14.
15. class _GroupDialogState extends State<GroupDialog> {
16.   late TextEditingController _nameController;
17.   late String _selectedEmoji;
18.
19.   @override
20.   void initState() {
21.     super.initState();
22.     _nameController = TextEditingController();
23.     _selectedEmoji = '✉';
24.   }
25.
26.   @override
27.   void dispose() {
28.     _nameController.dispose();
29.     super.dispose();
30.   }
31.
32.   @override
33.   Widget build(BuildContext context) {
34.     return AlertDialog(
35.       shape: RoundedRectangleBorder(
36.         borderRadius: BorderRadius.circular(20),
37.       ),
38.       title: const Text('New Group'),
39.       content: Column(
40.         mainAxisSize: MainAxisSize.min,
41.         children: [
42.           GestureDetector(
43.             onTap: () async {
44.               final emoji = await _showEmojiPicker(_selectedEmoji);
```

```
45.          if (_emoji != null) {
46.              setState(() => _selectedEmoji = emoji);
47.          }
48.      },
49.      child: Container(
50.          width: 60,
51.          height: 60,
52.          decoration: BoxDecoration(
53.              color: const Color.fromARGB(255, 80, 79, 120).withOpacity(0.1),
54.              borderRadius: BorderRadius.circular(15),
55.          ),
56.          child: Center(
57.              child: Text(
58.                  _selectedEmoji,
59.                  style: const TextStyle(fontSize: 30),
60.              ),
61.          ),
62.      ),
63.  ),
64.  const SizedBox(height: 16),
65.  TextField(
66.      controller: _nameController,
67.      decoration: const InputDecoration(
68.          labelText: 'Group Name',
69.          border: OutlineInputBorder(),
70.      ),
71.      maxLength: 20,
72.  ),
73. ],
74. ),
75. actions: [
76.     TextButton(
77.         onPressed: () => Navigator.pop(context),
78.         child: const Text('Cancel'),
79.     ),
80.     ElevatedButton(
81.         onPressed: () async {
82.             if (_nameController.text.trim().isEmpty) return;
83.
84.             Navigator.pop(context);
85.             widget.onSave?.call(
86.                 _nameController.text.trim(),
87.                 _selectedEmoji,
88.             );
89.         },

```

```
90.        style: ElevatedButton.styleFrom(
91.            backgroundColor: const Color.fromARGB(255, 80, 79, 120),
92.            foregroundColor: Colors.white,
93.        ),
94.        child: const Text('Create'),
95.    ),
96. ],
97. );
98. }
99.
100. Future<String?> _showEmojiPicker(String currentEmoji) async {
101.     const emojis = [
102.         '❑', '❑', '❑', '❑', '❑', '❑', '❑', '❑', '❑', '❑', '',
103.         '❑', '❑', '❑', '❑', '❑', '❑', '❑', '❑', '❑', '❑', '',
104.         '❑', '❑', '❑', '❑', '❑', '❑', '❑', '❑', '❑', '❑', '',
105.     ];
106.
107.     return showDialog<String>(
108.         context: context,
109.         builder: (context) => Dialog(
110.             shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(20)),
111.             child: Padding(
112.                 padding: const EdgeInsets.all(20),
113.                 child: Column(
114.                     mainAxisSize: MainAxisSize.min,
115.                     children: [
116.                         const Text(
117.                             'Choose Emoji',
118.                             style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
119.                         ),
120.                         const SizedBox(height: 20),
121.                         SizedBox(
122.                             width: 300,
123.                             height: 200,
124.                             child: GridView.builder(
125.                                 gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
126.                                     crossAxisCount: 6,
127.                                     mainAxisSpacing: 10,
128.                                     crossAxisSpacing: 10,
129.                                 ),
130.                                 itemCount: emojis.length,
131.                                 itemBuilder: (context, index) {
132.                                     final emoji = emojis[index];
133.                                     return InkWell(
134.                                         onTap: () => Navigator.pop(context, emoji),
```

```
135.          child: Container(
136.            decoration: BoxDecoration(
137.              color: emoji == currentEmoji
138.                  ? const Color.fromARGB(255, 80, 79, 120).withOpacity(0.2)
139.                  : Colors.transparent,
140.              borderRadius: BorderRadius.circular(10),
141.            ),
142.            child: Center(
143.              child: Text(
144.                emoji,
145.                style: const TextStyle(fontSize: 24),
146.              ),
147.            ),
148.          ),
149.        );
150.      ],
151.    ),
152.  ),
153.  ],
154.  ),
155.  ),
156.  ),
157.  );
158. }
159. }
160.
161.
```

4.4.13 widgets.dart

```
1. // Export all widgets for easy importing
2. export 'event_card.dart';
3. export 'todo_card.dart';
4. export 'event_dialog.dart';
5. export 'todo_dialog.dart';
6. export 'group_dialog.dart';
7. export 'button_custom.dart';
8. export 'custom_dialog.dart';
9. export 'custom_link_text.dart';
10. export 'custom_text_field.dart';
11. export 'gradient_background.dart';
12. export 'main_navigation.dart';
13. export 'page_title.dart';
```

