รหัสน.ศ.64172110337-6
ชื่อ-สกุล นนทชัย โพธิ์ศรี
กลุ่มเรียน CPE64241A

# ใบงาน 9

## การโปรแกรมเซ็นเซอร์ระบุพิกัดดาวเทียม

วัตถุประสงค์ เพื่อให้นักศึกษาเข้าใจ และสามารถเขียนโปรแกรม Flutter ใช้งานเซ็นเซอร์ระบุพิกัดดาวเทียมบนอุปกรณ์บนโทรศัพท์เคลื่อนที่ สำหรับแสดงแผนที่ และสามารถสร้าง Application ที่ใช้งานได้ด้วยตนเอง

คำอธิบาย เขียนคำอธิบายส่วนต่าง ๆ ของ code โปรแกรมแต่ละบรรทัด และทำการปรับแก้ไขโปรแกรมตามกำหนด เอกสารค้นคว้าเพิ่มเติมจากเว็บไซต์ต้นฉบับ

https://console.cloud.google.com/

https://pub.dev/packages/geolocator

https://pub.dev/packages/google_maps_flutter

วิธีการ

1. Google Maps เป็นบริการของบริษัท Google และมี API ที่สามารถขอใช้สำหรับเชื่อมต่อกับ Application โปรแกรมที่สามารถสร้างขึ้นได้เอง เนื้อหาในใบงานนี้เป็นการฝึกเขียน Application เชื่อมต่อ API ของ Google เพื่อใช้บริการแผนที่ ปัจจุบันมีชื่อเรียกว่า Google Maps Platform
2. การขอใช้บริการดังกล่าวต้องทำผ่าน user account ของ Google (@gmail.com) ให้นักศึกษาล๊อกอินอีเมล อีเมลที่นักศึกษาใช้คือ
3. เปิดหน้าเว็ปไซต์ https://console.cloud.google.com/ เลือกหรือสร้าง project โดยคลิกที่ช่องตัวเลือกด้านขวาของคำว่า google cloud หากยังไม่มี project ให้สร้างใหม่ หากมี project อยู่แล้วจะใช้ก็ได้ ที่นักศึกษาใช้ project ชื่อว่าอะไร

4. เลือกหัวข้อ APIs & Services และต่อไปให้เลือกหัวข้อ Credentials ที่แถบเมนู ให้เลือก +Create Credentials เลือก API key นักศึกษาจะได้รับ API key เป็นหมายเลขใด

5. เปิดโปรแกรม VSC และ new project เพิ่ม code ต่อไปนี้ลงใน /android/app/src/main/AndroidManifest.xml วางในบรรทัดสุดท้ายก่อนจบ tag </manifest>

```
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

6. ทดสอบ run หากติดปัญหา error บันทึก error และวิธีการแก้ไข
   ข้อความ error ที่ได้รับ
   ไม่มีเออเร่อ
   นักศึกษาทำการแก้ไขอย่างไร

7. เพิ่ม code ต่อไปนี้ลงใน /android/app/src/main/AndroidManifest.xml
วางในบรรทัดสุดท้ายก่อนจบ tag </application>

```
<meta-data android:name="com.google.android.geo.API_KEY" android:value="YOUR KEY HERE"/>
```

จากนั้นแก้ไข android:value ให้เท่ากับรหัสหมายเลขที่ได้จากขั้นตอนการขอ API Key บันทึกบรรทัดเฉพาะบรรทัดที่ทำการแก้ไข

```
        <meta-data android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyD0d7LAcY5A-gUPzD9C_fW_2fyHTbiXpaU" />
```

8. ต่อไปเป็นขั้นตอนขอใช้งานอุปกรณ์ GPS ในตัวเครื่องโทรศัพท์มือถือ ให้นักศึกษาติดตั้ง package ชื่อ geolocator นักศึกษาใช้คำสั่งใด

flutter pub add geolocator

9. เมื่อติดตั้ง package ในข้อ 8 แล้วนักศึกษาต้องทำการ import อะไรลงใน code โปรแกรม

```
import 'package:geolocator/geolocator.dart';
```

10. การขอใช้ GPS เพื่อรับค่าตัวเลขพิกัดระบบแผนที่จากดาวเทียม นำมาแสดงบนจอภาพ นักศึกษาเลือกเขียนโปรแกรมในส่วน StatelessWidget หรือ StatefulWidget

StatefulWidget

11. ให้สร้างตัวแปร สำหรับเก็บค่าตำแหน่ง GPS โดยใช้ชนิดตัวแปรเป็น Position และให้ค่าเริ่มต้นเป็นค่า null นักศึกษาใช้คำสั่งอะไร

```
Position? currentPosition = null;
```

12. เพิ่ม method ต่อไปนี้ลงใน code ส่วนที่นักศึกษาเลือกจากข้อ 10

```
Future<dynamic> _getLocation() async {
    Position position = await Geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.high);
       ชื่อตัวแปรที่สร้างในข้อ 11 = position;
    return position;
  }
```

13. ทำไมจึงต้องใช้คำสั่ง async และ await

เพื่อรอรับตำแหน่งของเรา

14. ทำไมจึงต้องใช้คำสั่ง Future

เพราะว่าจะได้รับค่าในอนาคต

15. ให้อธิบายความหมายเมธอดในข้อ 12

ฟังก์ชัน _getLocation() นี้ใช้เพื่อดึงตำแหน่ง GPS ของอุปกรณ์ โดยใช้ Geolocator.getCurrentPosition() เพื่อดึงตำแหน่งที่มีความแม่นยำสูง เมื่อดึงตำแหน่งเสร็จ จะเก็บค่าไว้ในตัวแปรที่คุณสร้างไว้ (เช่น currentPosition) และส่งค่ากลับมา

16. ในส่วนของ Scaffold แอตทริบิวต์ body: ให้แก้ไข Code เป็น widget ต่อไปนี้ และทดสอบ run หากพบ error ให้แก้ไข

```
body:
      const Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text('Current Location:',),
          ]
        )
      ),
```

17. Code ในข้อ 16 เป็นการแสดงข้อความ 1 บรรทัดกลางจอ แต่เราได้เตรียม Column Widget ไว้แล้ว ให้เพิ่ม Code ส่วนนี้ลงไป ต่อจาก Text('Current Location:',),

```
FutureBuilder(
            future: _getLocation(),
            builder: (BuildContext context, AsyncSnapshot<dynamic> snapshot) =>
              snapshot.hasData
              ? const Text('show data')
              : const Text('no data'),
),
```

18. Code ส่วน FutureBuiler จะทำการสร้างส่วนของ widget ที่ได้กำหนดไว้ใน builder: หลังจากที่ได้รับข้อมูลจาก future: และเราได้สั่งให้เมธอด _getLocation() ทำงาน ดังนั้นเมื่อได้รับข้อมูลแล้วคำสั่งนี้จะสร้าง context ตอบกลับเป็น Text ซึ่งเราได้สร้างเงื่อนไขตรวจสอบโดยใช้ snapshot.hasData หากมีข้อมูลจะแสดง Widget Text('show data') แต่หากไม่ได้รับข้อมูลกลับมาจะแสดง widget Text('no data') เมื่อนักศึกษาทำความเข้าใจ Code ส่วนนี้แล้ว ให้ทดลอง run และบันทึกผลการ run โดยสังเกตการเปลี่ยนแปลงของสิ่งที่เกิดขึ้นบนจอภาพ


19. ให้แก้ไข Code จากการแสดงคำว่า 'no data' เป็นการแสดงภาพวงล้อเคลื่อนไหวเป็นวงกลมขณะรอข้อมูลอยู่ที่เคยใช้ในใบงานก่อนหน้านี้ ทดสอบ run และบันทึก code เฉพาะในส่วน FutureBuilder

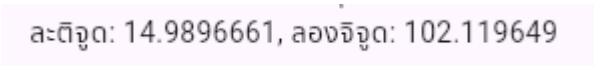
```
return const CircularProgressIndicator();
```

20. FutureBulder มีการเรียกใช้เมธอด _getLocation() และเมธอดได้ส่งค่าไปยังตัวแปรที่นักศึกษาสร้างขึ้นในข้อ 11 เราจะนำข้อมูลในตัวแปรตัวนั้น มาแสดงใน Text('show data') แทนคำว่า 'show data' อย่างไร ทดสอบ run และบันทึก code เฉพาะในส่วน FutureBuilder

```
return Text('ละติจูด: ${position?.latitude}, ลองจิจูด: ${position?.longitude}');
```

21. บันทึกภาพผลการ run ทั้งสองแบบ


22. รหัสพิกัดที่ได้มาเป็นพิกัด GPS ในสถานที่จริงขณะทำการทดลองด้วยอุปกรณ์มือถือที่รับสัญญาณจริงในขณะนี้ ให้บันทึกค่า Latitude และ Longitude ที่จับค่าได้




23. ในขั้นตอนต่อไปเป็นการนำแผนที่ในบริเวณที่ตรวจจับได้มาแสดงบน application ให้นักศึกษาติดตั้ง package ชื่อ google_maps_flutter นักศึกษาใช้คำสั่งใด

flutter pub add google_maps_flutter

24. เมื่อติดตั้ง package แล้วนักศึกษาต้อง import อะไรลงใน code

import 'package:google_maps_flutter/google_maps_flutter.dart';

25. เราจะเปลี่ยนจากการแสดงข้อความทั้งหมดของ body เป็นการแสดงแผนที่แทน ให้นำส่วนของ Center ออกจาก body และให้ FutureBuilder เป็นส่วนของ body แทน แก้ไข code แสดงภาพวงล้อเคลื่อนไหวเป็นวงกลมขณะรอข้อมูล อยู่กลางจอ บันทึก code ในส่วนของ body: FutureBuilder

```
body: FutureBuilder<Position>(
        future: _futurePosition,
        builder: (BuildContext context, AsyncSnapshot<Position> snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            // แสดงภาพวงล้อเคลื่อนไหวขณะรอข้อมูล
            return const Center(
              child: CircularProgressIndicator(),
            );
          } else if (snapshot.hasError) {
            return const Center(child: Text('เกิดข้อผิดพลาดในการดึงข้อมูล'));
          } else if (snapshot.hasData) {
            Position? position = snapshot.data;
            LatLng currentLatLng =
                LatLng(position!.latitude, position.longitude);

            return GoogleMap(
              onMapCreated: _onMapCreated,
              initialCameraPosition: CameraPosition(
                target: currentLatLng,
                zoom: 14.0, // ซูมแผนที่
              ),
```

```
                markers: {
                  Marker(
                    markerId: MarkerId("currentLocation"),
                    position: currentLatLng,
                    infoWindow: InfoWindow(title: "ตำแหน่งปัจจุบัน"),
                  ),
                },
              );
            } else {
              return const Center(child: Text('ไม่พบข้อมูลตำแหน่ง'));
            }
          },
        ),
```

26. สร้างตัวแปรต่อไปนี้

```
    GoogleMapController? mapController;
```

27. เพิ่มเมธอด _showMap() และเรียกใช้แทนการแสดงผล Latitude และ Longitude

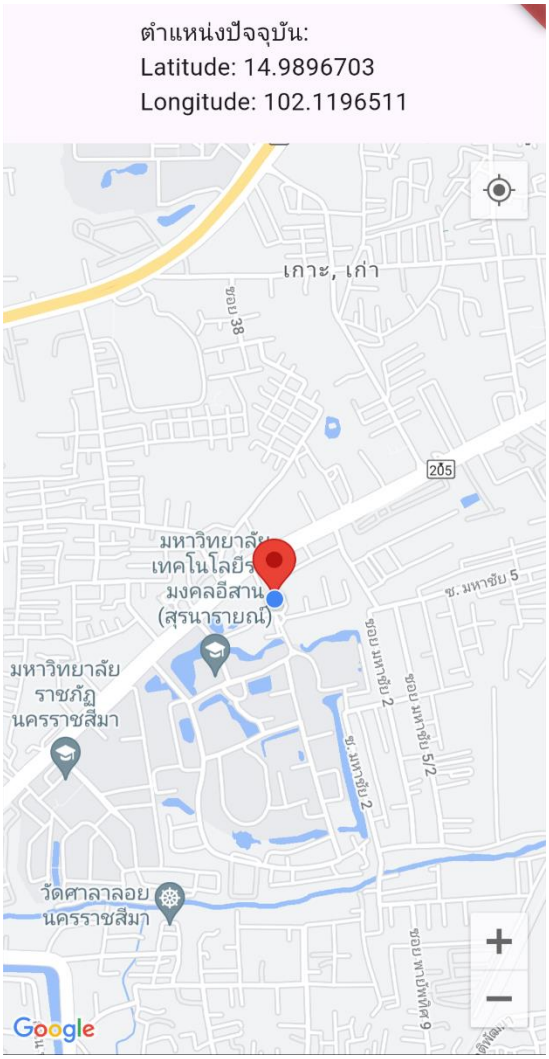
```
  _showMap(){
    return GoogleMap(
      mapType: MapType.normal,
      myLocationEnabled: true,
      onMapCreated: (GoogleMapController controller) => mapController = controller,
      initialCameraPosition: CameraPosition(
        target: LatLng(ชื่อตัวแปรที่สร้างในข้อ ๑!.latitude, ชื่อตัวแปรที่สร้างในข้อ ๑!.longitude),
        zoom: 15,
      ),
    );
  }
```

28. บันทึก code ส่วนของ body: FutureBuilder

```
body: FutureBuilder<Position>(
          future: _futurePosition,
          builder: (BuildContext context, AsyncSnapshot<Position> snapshot) {
            if (snapshot.connectionState == ConnectionState.waiting) {
              // แสดงภาพวงล้อเคลื่อนไหวขณะรอข้อมูล
              return const Center(
                child: CircularProgressIndicator(),
              );
            } else if (snapshot.hasError) {
              return const Center(child: Text('เกิดข้อผิดพลาดในการดึงข้อมูล'));
            } else if (snapshot.hasData) {
              Position? position = snapshot.data;
              LatLng currentLatLng =
                  LatLng(position!.latitude, position.longitude);

              return GoogleMap(
                mapType: MapType.normal,
                myLocationEnabled: true,
                onMapCreated: _onMapCreated,
                initialCameraPosition: CameraPosition(
                  target: currentLatLng,
```

```
              zoom: 15, // ซูมแผนที่
            ),
            markers: {
              Marker(
                markerId: const MarkerId("currentLocation"),
                position: currentLatLng,
                infoWindow: const InfoWindow(title: "ตำแหน่งปัจจุบัน"),
              ),
            },
          );
        } else {
          return const Center(child: Text('ไม่พบข้อมูลตำแหน่ง'));
        }
      },
    ),
```

29. ทดสอบ run และบันทึกผลจอภาพขณะทำงาน



30. นักศึกษาเป็นแผนที่และ ตำแหน่งพิกัดปัจจุบันบนจอภาพใช่หรือไม่

ใช่ค้าบ

31. หากเห็นตำแหน่งพิกัดปัจจุบัน เป็นพิกัดที่แสดงถูกต้องหรือไม่

ถูกต้องค้าบ

32. ทดสอบปรับแอตทริบิวต์ zoom เป็นค่าตัวเลขอื่น และอธิบายความหมายของตัวเลขในส่วนนี้

**ค่า zoom ที่ต่ำ (เช่น zoom: 5):**

จะทำให้แผนที่แสดงภาพที่กว้างขึ้นและดูเหมือนว่าผู้ใช้กำลังมองจากที่สูงขึ้น

เหมาะสำหรับการดูภาพรวมของภูมิภาคหรือเมืองใหญ่

**ค่า zoom ปานกลาง (เช่น zoom: 10):**

จะให้รายละเอียดของพื้นที่มากขึ้น เช่น ดูท้องถิ่นของเมืองหรือเขตที่เฉพาะเจาะจง

เหมาะสำหรับการมองดูละแวกใกล้เคียงหรือพื้นที่ที่ต้องการรายละเอียดพอสมควร

**ค่า zoom ที่สูง (เช่น zoom: 15 หรือสูงกว่า):**

จะทำให้แผนที่แสดงรายละเอียดมากขึ้น เช่น ถนน หรืออาคาร

เหมาะสำหรับการดูพื้นที่ที่เฉพาะเจาะจง เช่น แผนที่ในเมืองหรือพื้นที่ที่ต้องการดูรายละเอียดสูง

33. หากปรับค่าแอตทริบิวต์ myLocationEnabled เป็น false จะมีผลเป็นอย่างไร

เมื่อ myLocationEnabled ถูกตั้งค่าเป็น false, ปุ่มที่ให้ผู้ใช้สามารถดูตำแหน่งปัจจุบันของตนเองจะไม่ปรากฏบนแผนที่

34. แอตทริบิวต์ mapType สามารถปรับเปลี่ยนเป็นแบบใดได้บ้าง และแต่ละแบบใช้แสดงผลแผนที่ลักษณะใด

**1. MapType.normal**

ลักษณะ: แผนที่พื้นฐานแบบที่ผู้ใช้คุ้นเคย ซึ่งแสดงถนน, สถานที่, และข้อมูลพื้นฐาน

การใช้งาน: ใช้สำหรับแผนที่ทั่วไปที่ต้องการข้อมูลแผนที่แบบพื้นฐานและชัดเจน

**2. MapType.satellite**

ลักษณะ: แผนที่ภาพถ่ายดาวเทียมที่แสดงภาพจริงจากอากาศ

การใช้งาน: ใช้เมื่อคุณต้องการให้ผู้ใช้เห็นภาพถ่ายดาวเทียมของพื้นที่จริง เพื่อดูรายละเอียดเชิงลึกของพื้นที่ เช่น ลักษณะภูมิประเทศหรือโครงสร้าง

**3. MapType.hybrid**

ลักษณะ: แผนที่ที่รวมกันระหว่างแผนที่พื้นฐานและภาพถ่ายดาวเทียม โดยแสดงข้อมูลถนนและสถานที่บนพื้นฐานของภาพถ่ายดาวเทียม

การใช้งาน: ใช้เมื่อคุณต้องการข้อมูลที่รวมทั้งแผนที่พื้นฐานและภาพถ่ายดาวเทียม เพื่อให้การแสดงผลเป็นเชิงลึกมากขึ้นและเข้าใจง่าย

**4. MapType.terrain**

ลักษณะ: แผนที่ที่แสดงลักษณะภูมิประเทศ เช่น ภูเขา, หุบเขา, และลักษณะพื้นผิวของโลก

การใช้งาน: ใช้เมื่อคุณต้องการแสดงรายละเอียดของภูมิประเทศหรือการเดินทางที่ต้องคำนึงถึงลักษณะทางภูมิศาสตร์

35. บันทึก code ทั้งหมด

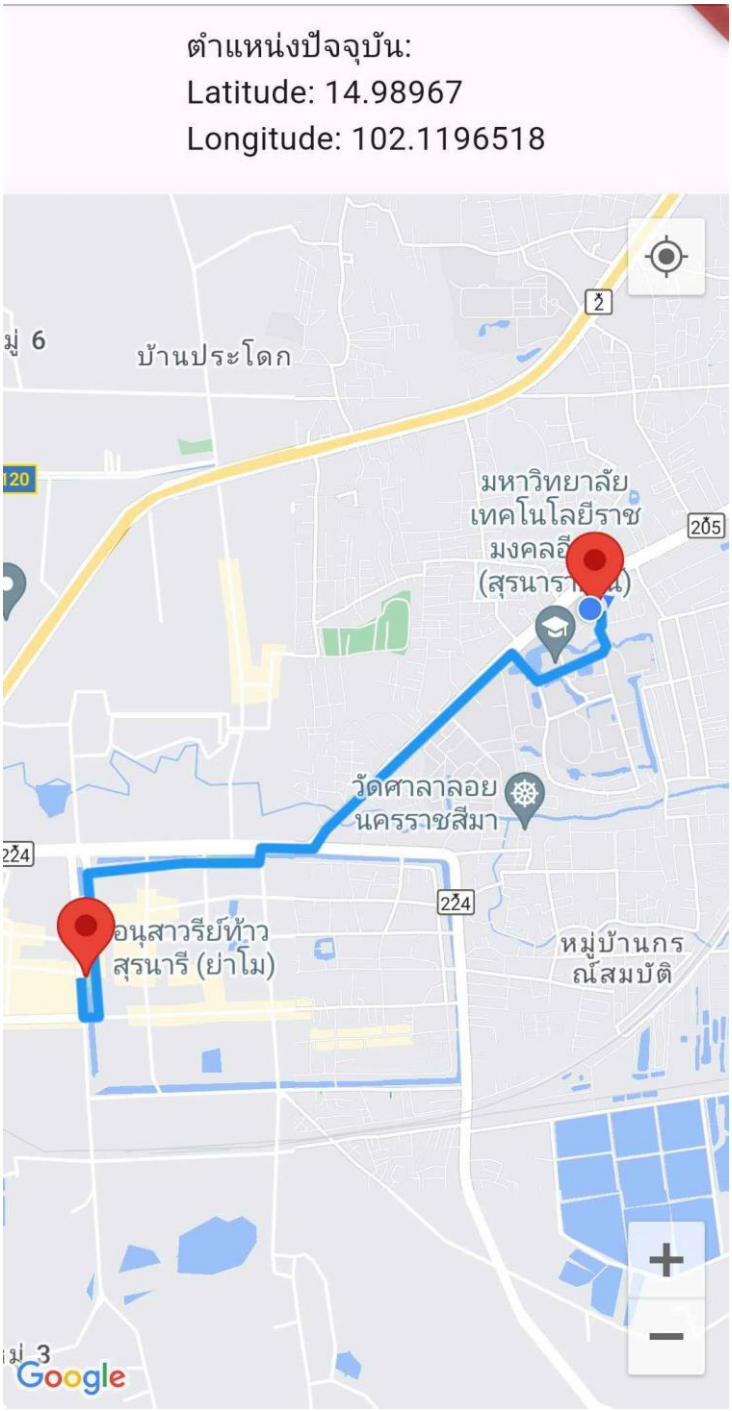
```
import 'package:flutter/material.dart';
import 'package:geolocator/geolocator.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:connectivity_plus/connectivity_plus.dart';
```

```dart
void main() {
  runApp(const MainApp());
}

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

  @override
  _MainAppState createState() => _MainAppState();
}

class _MainAppState extends State<MainApp> {
  Future<Position>? _futurePosition;
  GoogleMapController? _mapController;
  Position? currentPosition;
  bool isConnected = true;

  @override
  void initState() {
    super.initState();
    _checkConnection();
  }

  Future<void> _checkConnection() async {
    var connectivityResult = await Connectivity().checkConnectivity();
    if (connectivityResult == ConnectivityResult.none) {
      setState(() {
        isConnected = false;
      });
    } else {
      setState(() {
        isConnected = true;
        _futurePosition = _getLocation();
      });
      await _checkLocationPermission(); // ตรวจสอบสิทธิ์ตำแหน่งเมื่อมีการเชื่อมต่ออินเทอร์เน็ต
    }
  }

  Future<void> _checkLocationPermission() async {
    bool serviceEnabled;
    LocationPermission permission;

    serviceEnabled = await Geolocator.isLocationServiceEnabled();
    if (!serviceEnabled) {
      await Geolocator.openLocationSettings();
      return;
    }
```

```
  permission = await Geolocator.checkPermission();
  if (permission == LocationPermission.denied) {
    permission = await Geolocator.requestPermission();
    if (permission == LocationPermission.denied) {
      return;
    }
  }

  if (permission == LocationPermission.deniedForever) {
    return;
  }
}

Future<Position> _getLocation() async {
  return await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.high);
}

void _onMapCreated(GoogleMapController controller) {
  _mapController = controller;
  if (currentPosition != null) {
    _mapController?.animateCamera(
      CameraUpdate.newLatLng(
        LatLng(currentPosition!.latitude, currentPosition!.longitude),
      ),
    );
  }
}

@override
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      body: isConnected
          ? FutureBuilder<Position>(
              future: _futurePosition,
              builder:
                  (BuildContext context, AsyncSnapshot<Position> snapshot) {
                if (snapshot.connectionState == ConnectionState.waiting) {
                  return const Center(child: CircularProgressIndicator());
                } else if (snapshot.hasError) {
                  return const Center(
                      child: Text('เกิดข้อผิดพลาดในการดึงข้อมูล'));
                } else if (snapshot.hasData) {
                  Position position = snapshot.data!;
                  LatLng currentLatLng =
                      LatLng(position.latitude, position.longitude);
```

```
                    return Column(
                      children: [
                        const SizedBox(
                          height: 30,
                        ),
                        Padding(
                          padding: const EdgeInsets.all(16.0),
                          child: Text(
                            'ตำแหน่งปัจจุบัน:\nLatitude:
${position.latitude}\nLongitude: ${position.longitude}',
                            style: const TextStyle(fontSize: 16),
                          ),
                        ),
                        Expanded(
                          child: GoogleMap(
                            mapType: MapType.normal,
                            myLocationEnabled: true,
                            myLocationButtonEnabled: true,
                            onMapCreated: _onMapCreated,
                            initialCameraPosition: CameraPosition(
                              target: currentLatLng,
                              zoom: 15,
                            ),
                            markers: {
                              Marker(
                                markerId: const MarkerId("currentLocation"),
                                position: currentLatLng,
                                infoWindow:
                                    const InfoWindow(title: "ตำแหน่งปัจจุบัน"),
                              ),
                            },
                          ),
                        ),
                      ],
                    );
                  } else {
                    return const Center(child: Text('ไม่พบข้อมูลตำแหน่ง'));
                  }
                },
              )
            : const Center(
                child: Text('กรุณาเชื่อมต่ออินเทอร์เน็ตเพื่อใช้งานแอปนี้'')),
      ),
    );
  }
}
```

36. ให้แก้ไข code ให้สามารถค้นหาเส้นทางจากสถานที่ปัจจุบัน ไปยังสถานที่ปลายทางคือ อนุสาวรีย์ท้าวสุรนารี แสดงภาพเส้นทางบนแผนที่ บันทึก code และบันทึกผลจอภาพขณะทำงาน




```dart
import 'package:flutter/material.dart';
import 'package:geolocator/geolocator.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:connectivity_plus/connectivity_plus.dart';
import 'package:flutter_polyline_points/flutter_polyline_points.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
```

```dart
void main() {
  runApp(const MainApp());
}

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

  @override
  _MainAppState createState() => _MainAppState();
}

class _MainAppState extends State<MainApp> {
  Future<Position>? _futurePosition;
  GoogleMapController? _mapController;
  Position? currentPosition;
  bool isConnected = true;
  Set<Polyline> _polylines = {};
  List<LatLng> polylineCoordinates = [];
  final LatLng destinationLatLng = LatLng(14.9747, 102.0981);

  @override
  void initState() {
    super.initState();
    _checkConnection();
  }

  Future<void> _checkConnection() async {
    var connectivityResult = await Connectivity().checkConnectivity();
    if (connectivityResult == ConnectivityResult.none) {
      setState(() {
        isConnected = false;
      });
    } else {
      setState(() {
        isConnected = true;
        _futurePosition = _getLocation();
      });
      await _checkLocationPermission();
    }
  }

  Future<void> _checkLocationPermission() async {
    bool serviceEnabled;
    LocationPermission permission;

    serviceEnabled = await Geolocator.isLocationServiceEnabled();
    if (!serviceEnabled) {
      await Geolocator.openLocationSettings();
```

```
    return;
  }

  permission = await Geolocator.checkPermission();
  if (permission == LocationPermission.denied) {
    permission = await Geolocator.requestPermission();
    if (permission == LocationPermission.denied) {
      return;
    }
  }

  if (permission == LocationPermission.deniedForever) {
    return;
  }
}

Future<Position> _getLocation() async {
  return await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.high);
}

void _onMapCreated(GoogleMapController controller) {
  _mapController = controller;
  if (currentPosition != null) {
    _mapController?.animateCamera(
      CameraUpdate.newLatLng(
        LatLng(currentPosition!.latitude, currentPosition!.longitude),
      ),
    );
  }
}

Future<void> _getDirections(LatLng origin, LatLng destination) async {
  String apiKey =
      'AIzaSyAjpYsSbYCewRVzIqwai5yuOTEafdzPFAgss'; // แทนที่ด้วย API Key ของคุณ
  String url = 'https://maps.googleapis.com/maps/api/directions/json'
      '?origin=${origin.latitude},${origin.longitude}'
      '&destination=${destination.latitude},${destination.longitude}'
      '&mode=driving' // หรือเปลี่ยนเป็น 'walking', 'bicycling', 'transit' ตามที่ต้องการ
      '&key=$apiKey';

  http.Response response = await http.get(Uri.parse(url));

  if (response.statusCode == 200) {
    var data = jsonDecode(response.body);
    print("API Response: $data");

    if (data['routes'] != null && data['routes'].isNotEmpty) {
```

```
      var points = data['routes'][0]['overview_polyline']['points'];
      print("Polyline Points: $points");

      List<PointLatLng> result = PolylinePoints().decodePolyline(points);
      print("Decoded Points: $result");

      polylineCoordinates.clear();
      result.forEach((PointLatLng point) {
        polylineCoordinates.add(LatLng(point.latitude, point.longitude));
      });

      print("Polyline Coordinates: $polylineCoordinates");

      setState(() {
        _polylines.clear();
        _polylines.add(
          Polyline(
            polylineId: PolylineId("route"),
            points: polylineCoordinates,
            color: Colors.blue,
            width: 5,
          ),
        );
      });
    } else {
      print('No routes found');
    }
  } else {
    print('Failed to load directions');
    print("Error: ${response.body}");
  }
}

@override
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      body: isConnected
          ? FutureBuilder<Position>(
              future: _futurePosition,
              builder:
                  (BuildContext context, AsyncSnapshot<Position> snapshot) {
                if (snapshot.connectionState == ConnectionState.waiting) {
                  return const Center(child: CircularProgressIndicator());
                } else if (snapshot.hasError) {
                  return const Center(
                      child: Text('เกิดข้อผิดพลาดในการดึงข้อมูล'));
                } else if (snapshot.hasData) {
```

```
                    Position position = snapshot.data!;
                    currentPosition = position;
                    LatLng currentLatLng =
                        LatLng(position.latitude, position.longitude);

                    // Get directions from current location to destination
                    _getDirections(currentLatLng, destinationLatLng);

                    return Column(
                      children: [
                        const SizedBox(height: 30),
                        Padding(
                          padding: const EdgeInsets.all(16.0),
                          child: Text(
                            'ตำแหน่งปัจจุบัน:\nLatitude:
${position.latitude}\nLongitude: ${position.longitude}',
                            style: const TextStyle(fontSize: 16),
                          ),
                        ),
                        Expanded(
                          child: GoogleMap(
                            mapType: MapType.normal,
                            myLocationEnabled: true,
                            myLocationButtonEnabled: true,
                            onMapCreated: _onMapCreated,
                            initialCameraPosition: CameraPosition(
                              target: currentLatLng,
                              zoom: 15,
                            ),
                            markers: {
                              Marker(
                                markerId: const MarkerId("currentLocation"),
                                position: currentLatLng,
                                infoWindow:
                                    const InfoWindow(title: "ตำแหน่งปัจจุบัน"),
                              ),
                              Marker(
                                markerId: const MarkerId("destination"),
                                position: destinationLatLng,
                                infoWindow: const InfoWindow(title: "ปลายทาง"),
                              ),
                            },
                            polylines:
                                _polylines, // Add the polyline to the map
                          ),
                        ),
                      ],
                    );
```

```
                } else {
                  return const Center(child: Text('ไม่พบข้อมูลตำแหน่ง'));
                }
              },
            )
          : const Center(
              child: Text('กรุณาเชื่อมต่ออินเทอร์เน็ตเพื่อใช้งานแอปนี้')),
      ),
    );
  }
}
```

การทำงานของโปรแกรม ☐ ถูกต้อง ☐ ไม่ถูกต้อง

นักเรียนมีความซื่อสัตย์ ต่อตนเอง และผู้อื่น พยายามทำด้วยตนเอง และไม่ส่งคำตอบให้ผู้อื่น เพื่อลดความพยายามศึกษาของผู้อื่น ไม่ขอ หรือรับคำตอบจากผู้อื่น เพื่อลดความสามารถของตนเอง แม้แต่ข้อเดียว

☐ มีความซื่อสัตย์ ☐ ไม่มีความซื่อสัตย์

นักเรียนสามารถทำใบงานฉบับนี้ได้ด้วยตนเอง โดยไม่ต้องถามผู้อื่น หรือคัดลอกใช่หรือไม่

☐ สามารถ ☐ ไม่สามารถ

หากนักเรียนไม่สามารถทำได้ด้วยตนเอง ให้เขียนสาเหตุ หรือเหตุผลที่ไม่สามารถทำได้