FLUTTER SPEECH APP

Taiwanese and Chinese Speech Synthesis
Taiwanese and Chinese Speech Recognition

SDK Set up

- path: android/app/build.gradle
- compilSdkVersion 31
- minSdkVersion 21
- targetVersion 31

Android API Levels

Version	SDK / API level	Version code	Codename	Cumulative usage [1]	Year
Android 12	Level 31	S	Snow Cone [2]	*No data*	2021
Android 11	Level 30	R	Red Velvet Cake [2]	33.3%	2020
	▪ `targetSdk` must be 30+ for new apps by August 2021 and app updates by November 2021.				
Android 10	Level 29	Q	Quince Tart [2]	61.9%	2019
	▪ `targetSdk` must now be 29+ for all app updates.				
Android 9	Level 28	P	Pie	76.2%	2018
Android 8	Level 27 Android 8.1	O_MR1	Oreo	83.9%	2017
	Level 26 Android 8.0	O		87.2%	
Android 7	Level 25 Android 7.1	N_MR1	Nougat	89.3%	2016
	Level 24 Android 7.0	N		92.4%	
Android 6	Level 23	M	Marshmallow	96.0%	2015
Android 5	Level 22 Android 5.1	LOLLIPOP_MR1	Lollipop	98.2%	2015
	Level 21 Android 5.0	LOLLIPOP, L		98.6%	2014
	▪ Jetpack Compose requires a `minSdk` of 21 or higher.				
Android 4	Level 19 [3] Android 4.4	KITKAT	KitKat	99.6%	2013

compileSdkVersion

1. 決定用哪一個 Android SDK 版本，來編譯你的應用程式。
2. 官方強烈推薦使用最新的SDK進行編譯

minSdkVersion

1. 應用程式可以運行版本的最低要求
2. Support Library 或 Google Play services，可能有他們自己的minSdkVersion

targetSdkVersion

1. 宣告你的 Android App 預期使用者在哪個版本使用最合適
2. 更改務必做全面性的測試。

Set Version

1. 綜合上面來看三者關係為:

```
minSdkVersion <=
targetSdkVersion <=
compileSdkVersion.
```

2. 理想上，在穩定狀態下三者的關係應該為：

```
minSdkVersion (lowest possible) <=
targetSdkVersion ==
compileSdkVersion (latest SDK)
```

For Mac User

- Specify iOS 10 as the target platform
- Modify /ios/Podfile first line

Project | README.md | flutter_tts.dart | main.dart | Info.plist | Podfile | pubspec.yaml

Flutter commands | Open iOS module

```
# Uncomment this line to define a global platform for your project
platform :ios, '10.0'

# CocoaPods analytics sends network stats synchronously affecting f
ENV['COCOAPODS_DISABLE_STATS'] = 'true'

project 'Runner', {
  'Debug' => :debug,
  'Profile' => :release,
  'Release' => :release,
}
```

pubspec.ymal

```yaml
environment:
  sdk: ">=2.14.0 <3.0.0"
```

```yaml
dependencies:
  flutter:
    sdk: flutter
  permission_handler: ^8.1.2
  flutter_sound_lite: ^8.1.9
  path_provider: ^2.0.7
  flutter_tts: ^3.2.4
```

Permission request

- Android:
  - Path:

  android/app/src/main/AndroidManifest.xml
  - Add:

  <uses-permission android:name="android.permission.RECORD_AUDIO" />
  - Location:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.speech_app">
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <application
```

Permission request (Contd.)

- IOS:

add microphone permission in /ios/Podfile

註解掉原本的

```
# post_install do |installer|
#   installer.pods_project.targets.each do |target|
#     flutter_additional_ios_build_settings(target)
#   end
# end
```

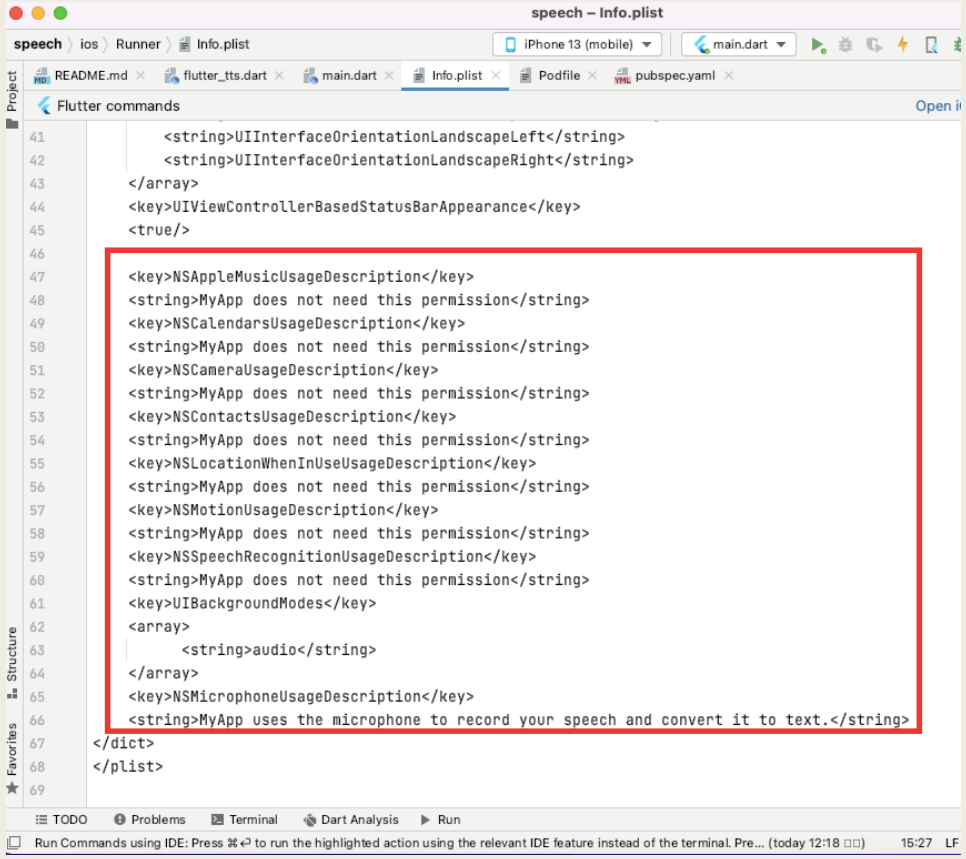
新增這一段

```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    flutter_additional_ios_build_settings(target)
    target.build_configurations.each do |config|
      config.build_settings['GCC_PREPROCESSOR_DEFINITIONS'] ||= [
        '$(inherited)',
        ## dart: PermissionGroup.microphone
        'PERMISSION_MICROPHONE=1',
      ]
    end
  end
end
```

•Reference: permission_handler -> setup -> ios

Permission request (Contd.)

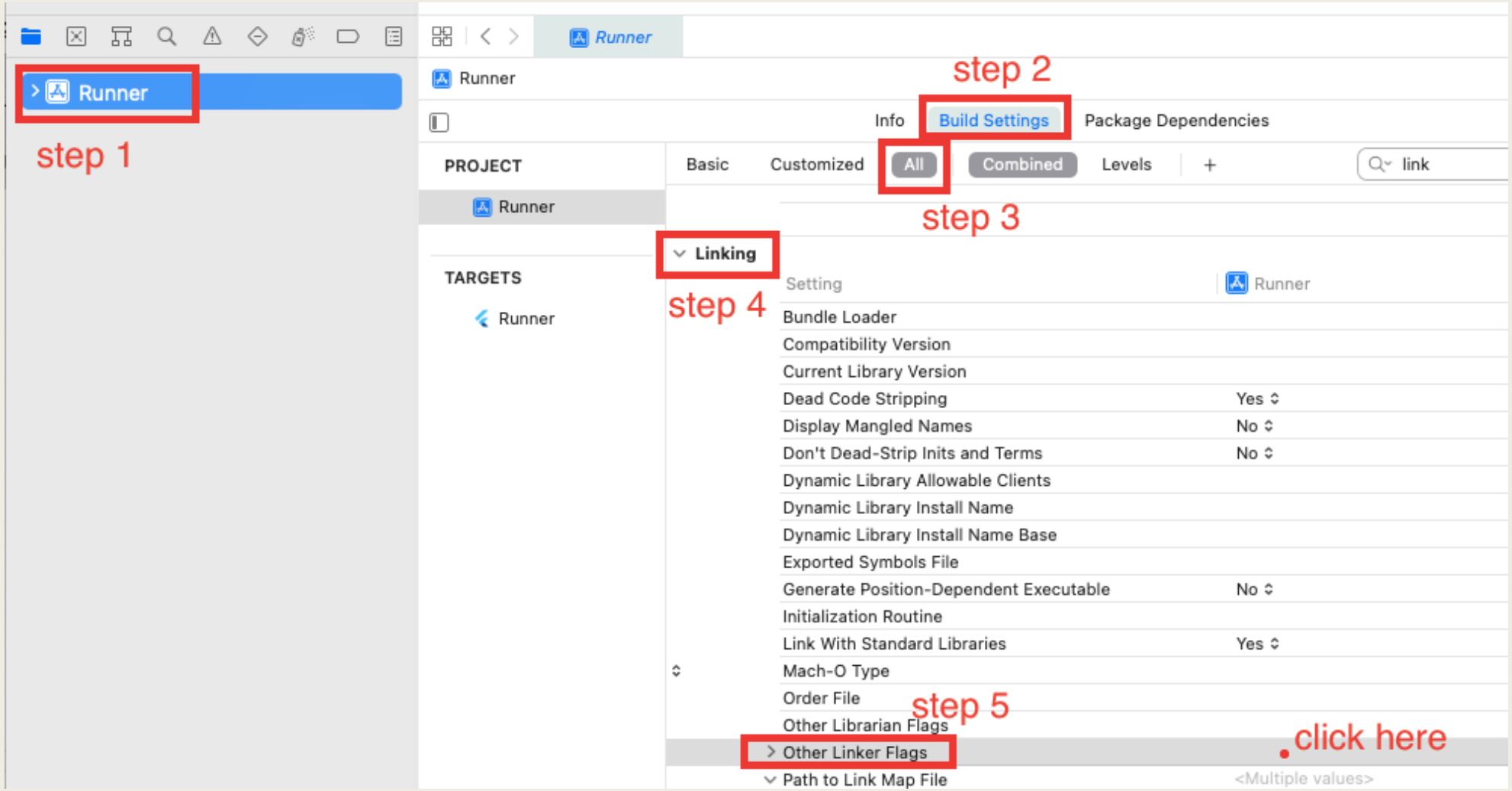
- IOS:
  - 在 /ios/Runner/info.plist 的 <dict> tag 裡新增權限請求代碼



For Mac User - linker flag

1. 用 Xcode 打開你的專案中的 ios 資料夾
2. 左方欄位點選 Runner
3. 右方點選 Build Settings -> All -> Linking -> other linker flag
4. 新增 flag: -lc++
5. 參考資料

For Mac User - linker flag (contd.)



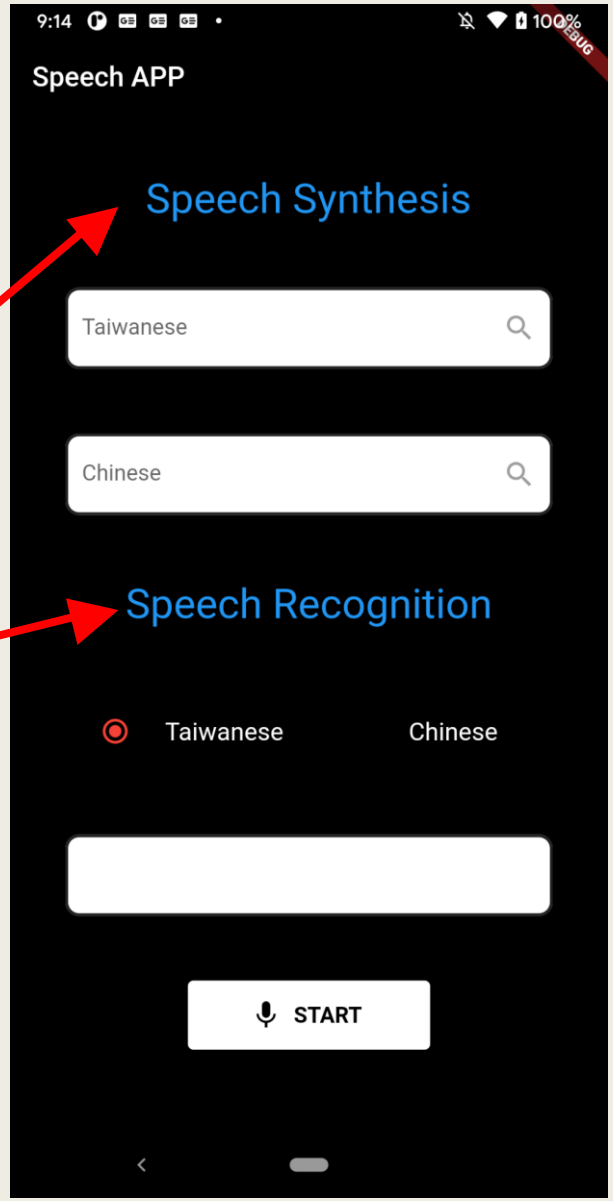
lib

- ios
- lib
  - flutter_tts.dart
  - generated_plugin_registrant.dart
  - main.dart
  - socket_stt.dart
  - socket_tts.dart
  - sound_player.dart
  - sound_recorder.dart
- test

main.dart

- Text
- TextField
- RadioListTile
- ElevatedButton

```
const Flexible(
  child: Center(
    child: Text(
      // content of text
      "Speech Synthesis",
      // Setup size and color of Text
      style: TextStyle(fontSize: 30, color: Colors.blue),
    ), // Text
  ), // Center
), // Flexible
```



main.dart (contd.)

```
      child: Text(
        // content of text
        "Speech Synthesis",
        // Setup size and color of Text
        style: TextStyle(fontSize: 30, color: Colors.blue),
      ), // Text
    ), // Center
  ), // Flexible
  Flexible(
      child: Center(
    child: buildTaiwaneseField("Taiwanese"),
  )), // Center // Flexible
  Flexible(
    child: Center(child: buildChineseField("Chinese")),
  ), // Flexible
  const Flexible(
      child: Center(
    child: Text(
      "Speech Recognition",
      style: TextStyle(fontSize: 30, color: Colors.blue),
    ), // Text
  )), // Center // Flexible
  Flexible(
    child: Center(child: buildRadio()),
  ), // Flexible
  Flexible(
    child: Center(child: buildOutputField()),
  ), // Flexible
  Flexible(
    child: Center(child: buildRecord()),
```

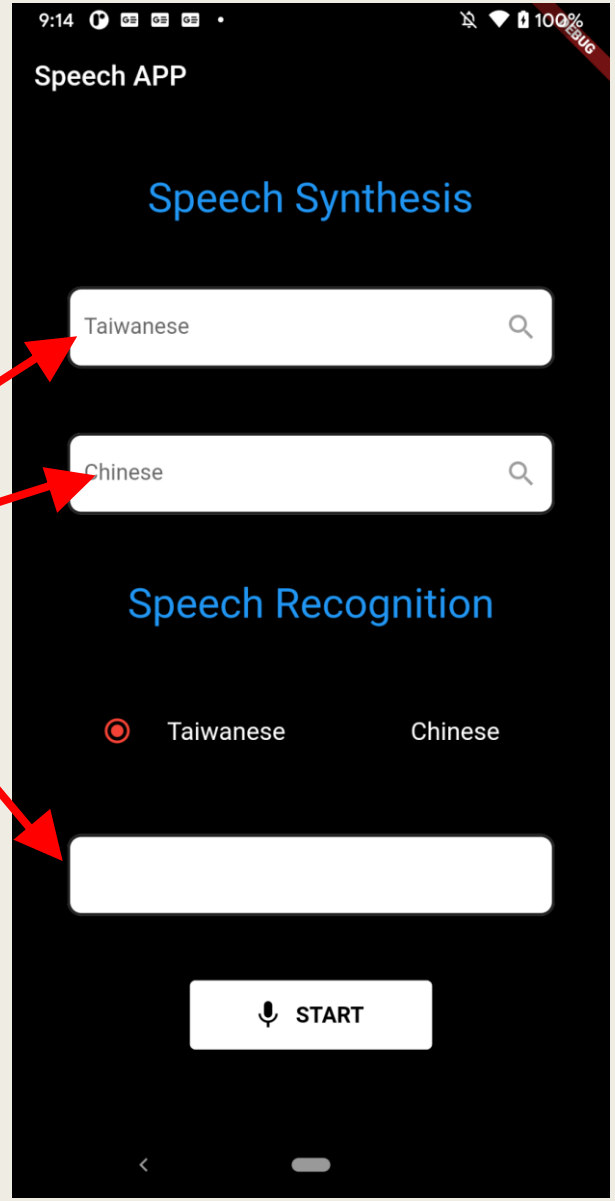
main.dart (contd.)

```
// get SoundRecorder
final recorder = SoundRecorder();
// get soundPlayer
final player = SoundPlayer();

// Declare TextEditingController to get the value in TextField
TextEditingController taiwanessController = TextEditingController();
TextEditingController chineseController = TextEditingController();
TextEditingController recognitionController = TextEditingController();
```

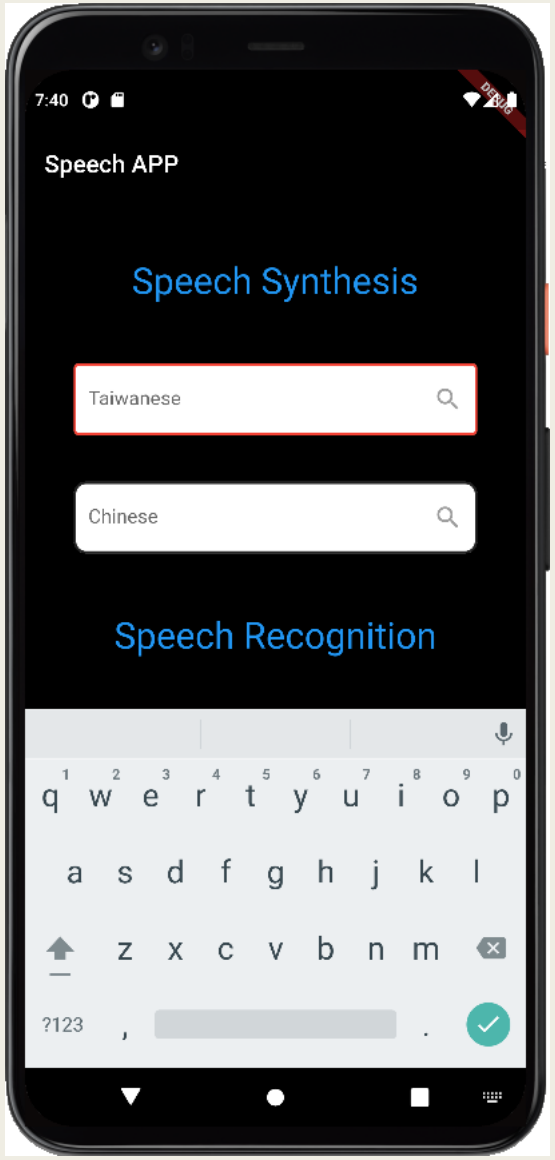
```
@override
void initState() {
  super.initState();
  recorder.init();
  player.init();
}
```

```
@override
void dispose() {
  recorder.dispose();
  player.dispose();
  super.dispose();
}
```

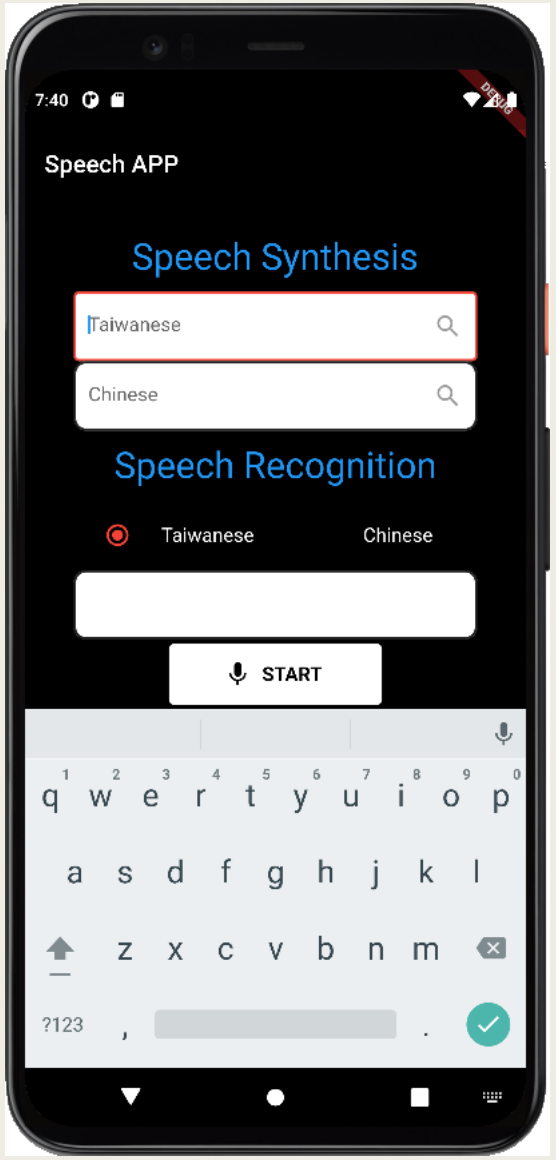


Set keyboard

```
@override
Widget build(BuildContext context) => Scaffold(
    // 設定不讓鍵盤技壓頁面
    resizeToAvoidBottomInset: false,
    appBar: AppBar(
        title: const Text('Speech APP'), backgroundColor: Colors.black), // AppBar
    // set background color
    backgroundColor: Colors.black,
```


7:40
Speech APP
Speech Synthesis
Taiwanese
Chinese
Speech Recognition

false


7:40
Speech APP
Speech Synthesis
Taiwanese
Chinese
Speech Recognition
Taiwanese
Chinese
START

true

TextField (input)

```
TextEditingController taiwanessController = TextEditingController();
```


```
Widget buildTaiwaneseField(txt) {
  return Padding(
    padding: const EdgeInsets.only(left: 40, right: 40),
    child: TextField(
      controller: taiwanessController,                          // 為了獲得TextField中的value
      decoration: InputDecoration(
        fillColor: Colors.white,                                // 背景顏色，必須結合filled: true,才有效
        filled: true,                                           // 重點，必須設定為true，fillColor才有效
        enabledBorder: const OutlineInputBorder(
          borderRadius: BorderRadius.all(Radius.circular(10)),  // 設定邊框圓角弧度
          borderSide: BorderSide(
            color: Colors.black87,                              // 設定邊框的顏色
            width: 2.0,                                         // 設定邊框的粗細
          ), // BorderSide
        ), // OutlineInputBorder
        // when user choose the TextField
        focusedBorder: const OutlineInputBorder(
            borderSide: BorderSide(
          color: Colors.red,                                    // 設定邊框的顏色
          width: 2,                                             // 設定邊框的粗細
        )), // BorderSide // OutlineInputBorder
        hintText: txt,                                          // 提示文字
        suffixIcon: IconButton(
          // TextField 中最後可以選擇放入 Icon
          icon: const Icon(
            Icons.search,                                        // Flutter 內建的搜尋 icon
            color: Colors.grey,                                  // 設定 icon 顏色
          ), // Icon
```

TextField (input) (Contd.)

Install: flutter pub add path_provide

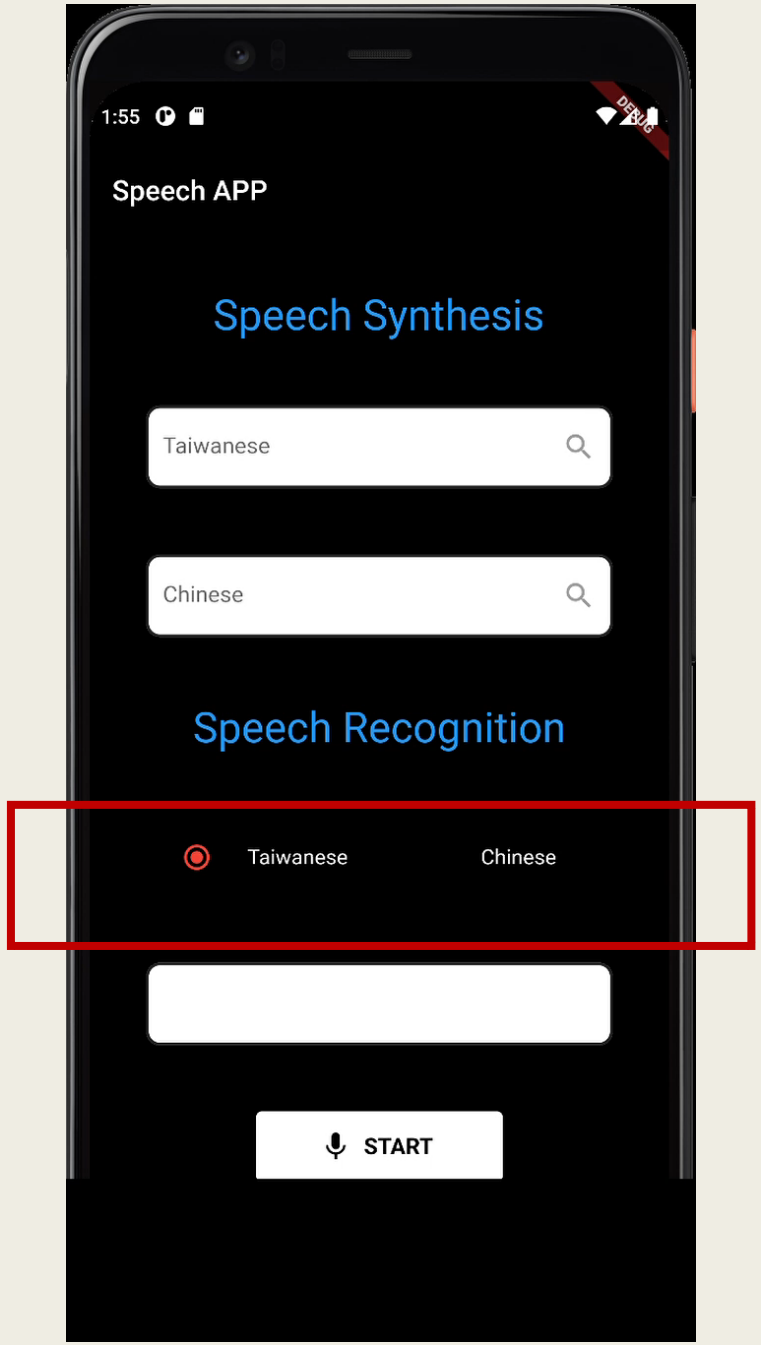
```
import 'package:path_provider/path_provider.dart' as path_provider;
import 'dart:io';
```

```
onPressed() async {
  // getTemporaryDirectory(): 取得暫存資料夾，這個資料夾隨時可能被系統或使用者操作清除
  Directory tempDir = await path_provider.getTemporaryDirectory();
  // define file path
  // String pathToReadAudio = '${tempDir.path}/example.wav';
  String pathToReadAudio = '${tempDir.path}/example.wav';
  // 得到 TextField 中輸入的 value
  String strings = taiwanessController.text;
  // 如果為空則 return
  if (strings.isEmpty) return;
  // connect to text2speech socket
  // The default is man voice.
  // If you want a female's voice, put "female" into the parameter.
  await Text2Speech()
      .connect(pathToReadAudio, player.play, strings, "female");
  // player.init();
  setState(() {});
```

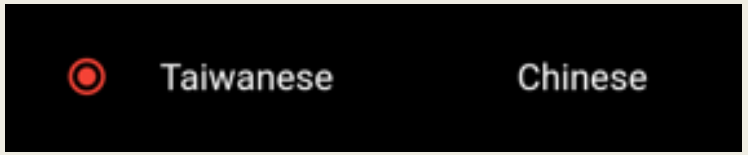
TextField (input)


```
onPressed: () async {
  String strings = chineseController.text;
  if (strings.isEmpty) return;
  print(strings);
  await Text2SpeechFlutter().speak(strings);
},
```

RadioListTile



RadioListTile



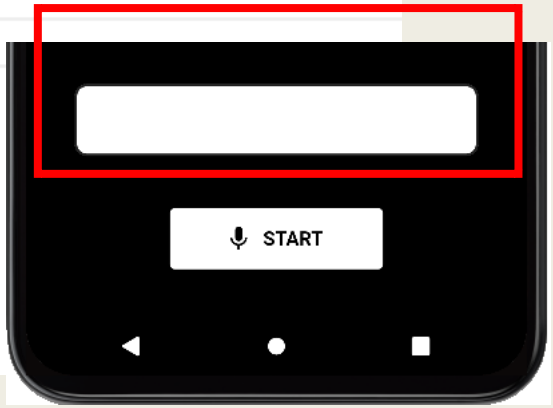

```
// Use to choose language of speech recognition
String recognitionLanguage = "Taiwanese";

Widget buildRadio() {
  return Row(children: <Widget>[
    Flexible(
      child: RadioListTile<String>(
        // 設定此選項 value
        value: 'Taiwanese',
        // Set option name、color
        title: const Text(
          'Taiwanese',
          style: TextStyle(color: Colors.white),
        ), // Text
        //  如果Radio的value和groupValu一樣就是此 Radio 選中其他設置為不選中
        groupValue: recognitionLanguage,
        // 設定選種顏色
        activeColor: Colors.red,
        onChanged: (value) {
          setState(() {
            // 將 recognitionLanguage 設為 Taiwanese
            recognitionLanguage = "Taiwanese";
          });
        },
      ), // RadioListTile
    ), // Flexible
```

```
Flexible(
  child: RadioListTile<String>(
    // 設定此選項 value
    value: 'Chinese',
    // Set option name、color
    title: const Text(
      'Chinese',
      style: TextStyle(color: Colors.white),
    ), // Text
    //  如果Radio的value和groupValu一樣就是此 Radio 選中其他設置為不選中
    groupValue: recognitionLanguage,
    // 設定選種顏色
    activeColor: Colors.red,
    onChanged: (value) {
      setState(() {
        // 將 recognitionLanguage 設為 Taiwanese
        recognitionLanguage = "Chinese";
      });
    },
  ), // RadioListTile
), // Flexible
```

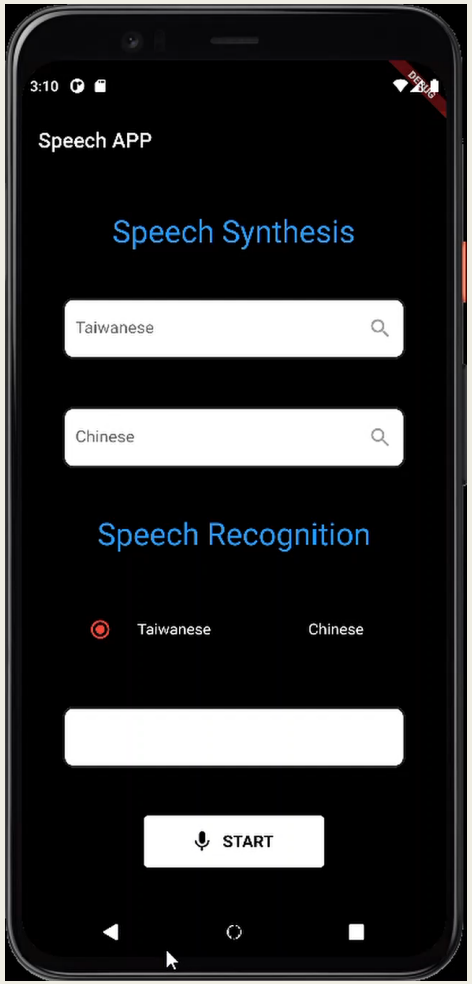
TextField (output)

```
Widget buildOutputField() {
  return Padding(
    padding: const EdgeInsets.only(left: 40, right: 40),
    child: TextField(
      controller: recognitionController,                    // 設定 controller
      enabled: false,                                       // 設定不能接受輸入
      decoration: const InputDecoration(
        fillColor: Colors.white,                            // 背景顏色，必須結合filled: true,才有效
        filled: true,                                       // 重點，必須設定為true，fillColor才有效
        disabledBorder: OutlineInputBorder(
          borderRadius: BorderRadius.all(Radius.circular(10)),  // 設定邊框圓角弧度
          borderSide: BorderSide(
            color: Colors.black87,                          // 設定邊框的顏色
            width: 2.0,                                     // 設定邊框的粗細
          ), // BorderSide
        ), // OutlineInputBorder
      ), // InputDecoration
    ), // TextField
  ); // Padding
}
```



ElevatedButton

```
// build the button of recorder
Widget buildRecord() {
  // whether is recording
  final isRecording = recorder.isRecording;
  // if recording => icon is Icons.stop
  // else => icon is Icons.mic
  final icon = isRecording ? Icons.stop : Icons.mic;
  // if recording => color of button is red
  // else => color of button is white
  final primary = isRecording ? Colors.red : Colors.white;
  // if recording => text in button is STOP
  // else => text in button is START
  final text = isRecording ? 'STOP' : 'START';
  // if recording => text in button is white
  // else => color of button is black
  final onPrimary = isRecording ? Colors.white : Colors.black;
```



ElevatedButton (Contd.)

```
import 'socket_stt.dart';
```

```
return ElevatedButton.icon(
  style: ElevatedButton.styleFrom(
    // 設定 Icon 大小及屬性
    minimumSize: const Size(175, 50),
    primary: primary,
    onPrimary: onPrimary,
  ),
  icon: Icon(icon),
  label: Text(
    text,
    // 設定字體大小及字體粗細（bold粗體，normal正常體）
    style: const TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
  ), // Text
  // 當 Iicon 被點擊時執行的動作
  onPressed: () async {
    // getTemporaryDirectory(): 取得暫存資料夾，這個資料夾隨時可能被系統或使用者操作清除
    Directory tempDir = await path_provider.getTemporaryDirectory();
    // define file directory
    String path = '${tempDir.path}/example.wav';
    // 控制開始錄音或停止錄音
    await recorder.toggleRecording(path, recognitionLanguage);
    // When stop recording, pass wave file to socket
    if (!recorder.isRecording) {
      if (recognitionLanguage == "Taiwanese") {
        // if recognitionLanguage == "Taiwanese" => use Minnan model
        await Speech2Text().connect(path, setTxt, "Minnan");
        // glSocket.listen(dataHandler, cancelOnError: false);
      } else {
        // if recognitionLanguage == "Chinese" => use MTK_ch model
        await Speech2Text().connect(path, setTxt, "MTK_ch");
      }
    }
    // set state is recording or stop
    setState(() {
      recorder.isRecording;
```

Call back function

```
// set recognitionController.text function
void setTxt(taiTxt) {
  setState(() {
    recognitionController.text = taiTxt;
  });
}
```

sound_recorder https://pub.dev/packages/flutter_sound_lite

Install: flutter pub add flutter_sound_lite

Use: import 'package:flutter_sound_lite/public/flutter_sound_recorder.dart';

```
// Declare FlutterSoundRecorder
FlutterSoundRecorder? _audioRecorder;
// Set recorder initislised is false
bool _isRecorderInitialised = false;
// isRecording => get status of recorder (whether is recording )
bool get isRecording => _audioRecorder!.isRecording;
```

sound_recorder(Contd.)

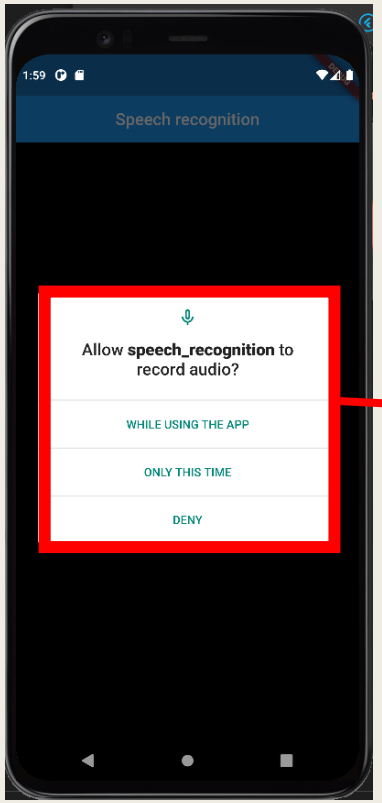
```
Non-nullable instance field '_audioRecorder' must be initialized.
Try adding an initializer expression, or a generative constructor that initializes it, or
mark it 'late'. dart(not_initialized_non_nullable_instance_field)
```

```
The property 'isRecording' can't be unconditionally accessed because the receiver can be
'null'.
Try making the access conditional (using '?.') or adding a null check to the target
('!'). dart(unchecked_use_of_nullable_value)
```

sound_recorder (Contd.)

Install: flutter pub add permission_handler

Use: import 'package:permission_handler/permission_handler.dart';




```dart
class SoundRecorder {
  // Declare FlutterSoundRecorder
  FlutterSoundRecorder? _audioRecorder;
  // Set recorder initislised is false
  bool _isRecorderInitialised = false;
  // isRecording => get status of recorder (whether is recording )
  bool get isRecording => _audioRecorder!.isRecording;

  Future init() async {
    // Get FlutterSoundRecorder
    _audioRecorder = FlutterSoundRecorder();
    // Get the permission status of microphone
    final status = await Permission.microphone.request();
    // If permission is deny => throw exception
    if (status != PermissionStatus.granted) {
      throw RecordingPermissionException('Micorphone permission denied');
    }
    // Open audiosession
    await _audioRecorder!.openAudioSession();
    // set recorder initislised is true
    _isRecorderInitialised = true;
  }
```

sound_recorder (Contd.)

```
// release recorder
void dispose() {
  // if Recorder isn't initialised => return
  if (!_isRecorderInitialised) return;
  // close audiosession
  _audioRecorder!.closeAudioSession();
  // set audiorecorder is null
  _audioRecorder = null;
  // set recorder initislised is true
  _isRecorderInitialised = false;
}
```

sound_recorder (Contd.)

```
// start recorder
Future _record(path) async {
  // if Recorder isn't initialised
  if (!_isRecorderInitialised) return;
  print('********* record outputpath : $path');
  // start recorder
  await _audioRecorder!.startRecorder(toFile: path);
}

// stop recorder
Future _stop() async {
  // if Recorder isn't initialised
  if (!_isRecorderInitialised) return;
  // stop recorder
  await _audioRecorder!.stopRecorder();
}
```

```
// Control the start or end of the recorder
// require parameter path and language of recognition
Future toggleRecording(path) async {
  // if recorder is stop
  if (_audioRecorder!.isStopped) {
    // start recorder
    await _record(path);
  } else {
    // else stop recorder
    await _stop();
  }
}
```

socket_stt

```
class Speech2Text {
  //若透過Android手機傳送，則設為"A"；若透過網頁傳送，則設為"W"
  final String label = "A";
  final serviceId = "0001";

  //由SERVER端提供之token
  final String token =
      "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzUxMiJ9.eyJpZCI6NzgsInVzZXJfaWQiOiIwIiwic2VydmljZV

  // Connect to socket
  // parameter: wav file path, call back function, and language
  Future connect(
      String path, void Function(String) handler, String language) async {
    String modelname = language + "\u0000\u0000";
    String outmsg = token + "@@@" + modelname + label + serviceId;

    //將outmsg轉成byte[]
    List<int> outmsgByte = utf8.encode(outmsg);
    //將語音檔案轉成byte[]，使用下方convert(String path) function
    List<int> waveByte = await convert(path);
    //將outmsg以及語音檔案兩個陣列串接，使用下方 byteconcate(byte[] a, byte[] b) function
    List<int> outbyte = byteconcate(outmsgByte, waveByte);

    //用於計算outmsg和語音檔案串接後的byte數
    var g = Uint32List(4);
    // little endian 轉 big endian
    g[0] = (outbyte.length & 0xff000000) >>> 24;
    g[1] = (outbyte.length & 0x00ff0000) >>> 16;
    g[2] = (outbyte.length & 0x0000ff00) >>> 8;
    g[3] = (outbyte.length & 0x000000ff);
```

```
import 'dart:typed_data';
```

socket_stt (Contd.)

import 'dart:convert';

Call back function

```
//連接socket
await Socket.connect("140.116.245.149", 2804).then((socket) {
  print('---------Successfully connected------------');
  // 向socket傳送資料
  socket.add(byteconcate(g, outbyte));
  socket.flush();
  // socket監聽
  socket.listen((dataByte) {
    print('-------Data from socket-------');
    // decode byte to string
    var dataString = utf8.decode(dataByte);
    // 因為dart中Map中的引號是雙引號,但回傳的json格式是單引號，所以會報錯=>需要轉換
    Map respone = jsonDecode(dataString.replaceAll("'", '"'));
    // 取得辨識結果中排名的第一名
    final taiTxt = respone['rec_result'][0];
    // 顯示回傳結果
    print(taiTxt);
    // 呼叫 call back function
    handler(taiTxt);
  });
  // catch error
}).catchError((e) {
  print("socket無法連接: $e");
});
}
```

socket_stt (Contd.)

```
import 'dart:io';
```

```
// 用於串接兩個byte[]
// List<int> = java 中的 byte[]
List<int> byteconcate(List<int> a, List<int> b) {
  // 宣告 result 為 size 是 (a.length + b.length) 的 sign 32bits 的 byte[]
  List<int> result = Int32List(a.length + b.length);

  /// Java的System.arrayCopy(source, sourceOffset, target, targetOffset, length)
  /// = target.setRange(targetOffset, targetOffset + length, source, sourceOffset);
  result.setRange(
      0, a.length, a, 0); // =System.arraycopy(a, 0, result, 0, a.length);
  result.setRange(a.length, a.length + b.length, b,
      0); // =System.arraycopy(b, 0, result, a.length, b.length);
  return result;
}

//用於將檔案轉換成byte，輸入為檔案路徑，輸出為byte[]
Future<List<int>> convert(path) async {
  // create file
  var file = File(path);
  // 以byte方式讀取檔案
  var bytes = await file.readAsBytes();
  return bytes;
}
```

socket_tts

```
// Connect to socket
// parameter: wav file path, call back function, speech synthesized text, (female)
// default model is man
Future connect(
    String pathToReadAudio, void Function(String) player, String strings,
    [String inputModel = 'man']) async {
  String model = inputModel;
  // choose man or female model
  if (inputModel == "man") {
    model = "M12_sandhi";
  } else if (inputModel == "female") {
    model = "F14_sandhi";
  }
```

Socket_tts (Contd.)

```
  // socket監聽
  socket.listen((dataByte) async {
    print('-------Data from socket-------');
    // get data from socket
    var data = dataByte;
    // close socket
    await socket.close();
    socket.destroy();
    // getTemporaryDirectory(): 取得暫存資料夾，這個資料夾隨時可能被系統或使用者操作清除
    Directory tempDir = await path_provider.getTemporaryDirectory();
    // define file path
    String pathToReadAudio = '${tempDir.path}/SpeechSynthesis.wav';
    print(data);
    // create file
    var file = File(pathToReadAudio);
    // write the data to file in byte
    await file.writeAsBytes(data, flush: true);
    // call back function
    player(pathToReadAudio);
  });
  // catch error
}).catchError((e) {
  print("socket無法連接: $e");
});
```

Flutter_tts

- Install: flutter pub add flutter_tts
- Pubspec.yaml: dependencies: flutter_tts: ^3.2.4
- Use: import 'package:flutter_tts/flutter_tts.dart’;
- https://pub.dev/packages/flutter_tts

flutter_tts (Contd.)

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

class Text2SpeechFlutter {
  final FlutterTts flutterTts = FlutterTts();
  Future speak(String strings) async {
    // print all language
    // print(await flutterTts.getLanguages);
    // set language
    await flutterTts.setLanguage("zh-TW");
    // set pitch
    // await flutterTts.setPitch();
    await flutterTts.speak(strings);
  }
}
```