APP & Node.js API Internal Guide

iOS APP

- **開發環境 :** macOS Catalina10.15.7, Xcode 12.4
- **開發語言 :** Swift 5
- **測試環境 :** iOS 13
- **Requirement :** UIKit, CoreLocation, MapKit, Alamofire = "5.2", SwiftyJSON
- **程式架構 :** AppDelegate.swift, SceneDelegate.swift, ViewController.swift, ChooseLabelTableViewController.swift, Main.storyboard
- **Code Description :**
  - ➢ **AppDelegate.swift** :

    class AppDelegate

    - ➔ **private func** application(application: UIApplication,didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool
    - ➔ **func** application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool
    - ➔ **func** application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration
    - ➔ **func** application(_ application: UIApplication, didDiscardSceneSessions sceneSessions: Set<UISceneSession>)
    - ➔ **func** applicationWillResignActive(_ application: UIApplication)
    - ➔ **func** applicationDidEnterBackground(_ application: UIApplication)
    - ➔ **func** applicationWillEnterForeground(_ application: UIApplication)
    - ➔ **func** applicationDidBecomeActive(_ application: UIApplication)
    - ➔ **func** applicationWillTerminate(_ application: UIApplication)

    AppDelegate 提供程式啟動、退出的介面，為建立應用程式之預設檔案。

➢ **SceneDelegate.swift** :

class SceneDelegate

➔ **func** scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions: UIScene.ConnectionOptions)

➔ **func** sceneDidDisconnect(_ scene: UIScene)

➔ **func** sceneDidBecomeActive(_ scene: UIScene)

➔ **func** sceneWillResignActive(_ scene: UIScene)

➔ **func** sceneWillEnterForeground(_ scene: UIScene)

➔ **func** sceneDidEnterBackground(_ scene: UIScene)

SceneDelegate 主要管理應用程式的 Scene，為建立應用程式之預設檔案。

➢ **ViewController.swift**:

class ViewController

➔ **override func** viewDidLoad()

對 APP 介面做初始化，包含產生相機與相簿的按鈕、設定各物件位置，以及開啟手機GPS 定位。

➔ **@objc func** onCameraBtnAction(_ sender: UIButton)

確認按下相機按鈕。

➔ **@objc func** onPhotoBtnAction(_ sender: UIButton)

確認按下相簿按鈕。

➔ **func** callGetPhoneWithKind(_ kind: Int)

根據上列兩個 function 傳入的參數決定開啟相機或相簿。

➔ **private func** imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String : Any])

進入相簿取得圖檔。

➔ **func** imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any])

拍照後 或 進入相簿取得圖檔後進入此 function。初始所有標籤欄位，並顯示預覽圖片，接著紀錄 "Auto Obtain" 的資料及呼叫 **func** analyzeResult

➔ **func** imagePickerControllerDidCancel(_ picker: UIImagePickerController)

PickerController 結束時呼叫。

➔ **func** locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation])
隨時記錄 "Auto Obtain" 資料中經度、緯度、高度和時間資訊。

➔ **func** locationManager(_ manager: CLLocationManager, didUpdateHeading orientation: CLHeading)
透過取得的方位角來產生"Auto Obtain" 資料中方位的資訊，每 22.5 度為一個方位。

➔ **func** analyzeResults(_ dataToParse: Data)
解析 Cloud Vision API 回傳的 JSON 檔，並記錄 Labels 與 Landmarks 的分析結果。

➔ **func** base64EncodeImage(_ image: UIImage) -> String 判斷照片大小決定是否呼叫 **func** resizeImage 並回傳 base64 encode 後的 結果。

➔ **func** resizeImage(_ imageSize: CGSize, image: UIImage) -> Data
將照片轉為 png 的格式並壓縮至 2MB 以下以方便傳輸。

➔ **@objc func** closeKeyboard()
輸入完成後呼叫此 function 以關閉鍵盤。

➔ **@IBAction func** done(_ sender: Any)
將 使用者輸入(Manual Enter)、Auto Obtain、使用者選取的 Labels 與 Landmarks(Cloud Vision) 組合成 JSON 的格式，接著使用 AlamoFire 傳送 至 Node.js API，同時將照片傳至 Server Photo Database

extension ViewController : UIPickerViewDataSource

➔ **func** numberOfComponents(in pickerView: UIPickerView) -> Int

➔ **func** pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int
計算 範疇 及 喜好程度 下拉式選單的數量

extension ViewController : UIPickerViewDelegate

➔ **func** pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String?
設定 範疇 及 喜好程度 下拉式選單的內容

➔ **func** pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int)
顯示 範疇 及 喜好程度 下拉式選單的使用者所選內容

➢ **ChooseLabelTableViewController.swift** :
class ChooseLabelTableViewController

➔ **override func** viewDidLoad()
初始化 Labels 與 Landmarks 選取畫面的 ViewController。

➔ **override func** numberOfSections(in tableView: UITableView) -> Int
在 TableView 中設定 2 個 section。

➔ **override func** tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String?
設定 2 個 section 的標題，分別為 Labels 與 Landmarks。

➔ **override func** tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
計算 Labels 與 Landmarks 區塊的數量。

➔ **override func** tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
設定 Labels 與 Landmarks 區塊的內容，並預設勾選全部項目。

➔ **override func** tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath)
紀錄使用者所勾選的 Labels 與 Landmarks 項目

➢ **Main.storyboard**:
由此添加畫面上需顯示的 object，並將 object 拉至對應的的 class 形成可操作的 variable。

Android APP

- **開發環境 :** Windows 10 1909, Android Studio 4.0.1
- **開發語言 :** Java1.8.0_242
- **測試環境 :** Android 9.0
- **Requirement** : junit:junit:4.12 , com.android,support:appcompat-v7:27.0.2,
  com.android.support:design:27.0.2 ,
  com.google.api-client:google-api-client-android:1.23.0
  com.google.http-client:google-http-client-gson:1.23.0
  com.google.apis:google-api-services-vision:v1-rev369-1.23.0
  com.android.volley:volley:1.1.1
- **程式架構 :** MainActivity.java, MulAdapter.java, LandmarkListAdapter,java,
  PermissionUtils.java, PackageManagerUtils.java,
  VolleyMultipartRequest.java, activity_main.xml, content_main.xml,
  label_choose_dialog.xml, listview_checkbox.xml,
  listview_checkbox2.xml
- **Code Description ：**
  - **MainActivity.java**

    class MainActivity
    - ➔ **override func** onCreate(Bundle savedInstanceState)
      程式啟動後要執行的動作
    - ➔ **func** findObject()
      將程式中的變數與物件連結起來
    - ➔ **func** setOnClickEvent()
      設置程式中物件的點擊事件
    - ➔ **func** init()
      初始化程式中的變數
    - ➔ **func** startGalleryChooser()
      選擇從裝置中選取照片後的動作
    - ➔ **func** startCamera()
      選擇拍照後的動作
    - ➔ **func** getCamerafile()
      獲得照片所在資料夾的位置名稱

➔ **override func** onActivityResult(int requestCode, int resultcode, Intent data)

func startActivityForResult 後執行的動作

➔ **override func** onRequestPermissionResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults)

接收使用者回應以確認權限的開啟

➔ **func** uploadImg(Uri uri)

上傳照片給 Cloud Vision

➔ **func** prepareAnnotationRequest(Bitmap bitmap)

設定 Cloud Vision 的請求註解，如要回傳哪些項目與數量等等

class LabelDetectionTask

➔ 設定請求 Cloud Vision 前後要執行的動作

- ❖ **func** doInBackground(Object… params)

  請求 Cloud Vision 後，在背景執行的動作
- ❖ **func** onPostExecute(String result)

  收到 Cloud Vision 傳回的結果後執行的動作

➔ **func** callCloudVision(final Bitmap bitmap)

請求 Cloud Vision

➔ **func** scaleBitmapDown(Bitmap bitmap, int maxDimension)

壓縮圖片大小

➔ **func** convertResponseToString(BatchAnnotatrImagesResponse response)

將 Cloud Vision 回傳的結果轉換為我們需要的字串

➔ **SensorEventListener** acc_listener

設定 sensor 的事件

➔ **func** onLocationChanged(Location location)

設定座標發生變動時的事件

➔ **func** getData()

取得我們需要的資料

➔ **func** uploadBitmap(final Bitmap bitmap)

上傳照片到 server

➔ **func** getJSONObject()

將資料製作成 json

➔ **func** uploadJson(JSONObject object)

上傳 json 到 database

➢ **LandmarkListAdapter.java**

class LandmarkListAdapter

➔ **func** LandmarkListAdapter(Activity a, List<String> list)

➔ **func** getCount()

➔ **func** getItem()

➔ **func** getItemId()

➔ **func** getView()

LandmarkListAdapter 為用於建立給使用者進行多選的 Listview 所建立的 class

➢ **MulAdapter.java**

class MulAdapter

作用與內容等同於 LandmarkListAdapter，僅用於區隔兩個不同的 Listview

➢ **PackageManagerUtils.java**

class PackageManagerUtils

獲取 Google Cloud Vision API 時所需使用的 Signature

➢ **PermissionUtils.java**

class PermissionUtils

➔ **func** requestPermission(Activity activity, int requestCode, String… permissions)

➔ **func** permissionGranted(int requestCode, int permissionCode, int[] grantResults)

用於判斷權限是否開通與請求權限

➢ **VolleyMultipartRequest.java**

class VolleyMultipartRequest

設定 volley 上傳圖片的相關設定

➢ **activity_main.xml**

程式主畫面的版面配置

➢ **content_main.xml**

配置程式主畫面中的 object 及其位置

➢ **label_choose_dialog.xml**

彈出式視窗的版面配置及其 object

➢ **listview_checkbox.xml**

給 listview 用的 checkbox

➢ **listview_checkbox2.xml**

等同於 listview_checkbox.xml，只是配置給不同的 listview

Node.js API

- **開發環境 :** Windows 10 1909
- 開發語言 : Node.js 14.16.0
- **測試環境 :** 8.0.23 MySQL Community Server - GPL, iOS 13, Android 9.0
- **Requirement** : mysql = “2.18.1”, http = “0.0.1”
- **程式架構 :** mysql.createConnection, http.createServer
- **Code Description** :
  - ➢ **mysql.createConnection** :
    使用 mysql 套件的 createConnection 連接至 MySQL Database
  - ➢ **http.createServer** :
    接收 APP 所傳之 JSON 檔，解析後將內容分別放入MySQL Database 的 photo_tags 和 vision_api 兩張資料表之中。