致理科技大學

商務科技管理系

實 務 專 題 報 告

題 目

捷出高手－北捷出口指引 APP

指導老師：林正平

學生：廖翊凱（10533207）

余姿嬋（10533228）

林季穎（10533229）

黃玟綺（10533237）

蔡秉翰（10533208）

中華民國 108 年 12 月

致理科技大學

商務科技管理系

實 務 專 題 報 告

題 目

捷出高手－北捷出口指引 APP

學生：廖翊凱（10533207）

余姿嬋（10533228）

林季穎（10533229）

黃玟綺（10533237）

蔡秉翰（10533208）

本成果報告書經審查及口試合格特此證明

指導老師（親簽）：_____

中華民國 108 年 12 月

CTM 實務專題研究授權書

本授權書所授權之實務專題研究為

_____ 共 _____ 人，

在致理科技大學商務科技管理系_____學年度第_____學期完成商管實務專題。

商管實務專題名稱：

_____

□同意 □ 不同意 本組同學共 _____ 人，皆同意著作財產權之論文全文資料，授予

教育部指定送繳之圖書館及本人畢業學校圖書館，為學術研究之目的以各種方法重製，或為上述目的在授權他人以各種方法重製，不限地域與時間，惟每人以一份為限。

上述授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。上述同意與不同意之欄位若未勾選，該組同學皆同意視同授權。

指導教授姓名(親筆正楷)：

專題生簽名(親筆正楷)： 學號：

專題生簽名(親筆正楷)： 學號：

專題生簽名(親筆正楷)： 學號：

專題生簽名(親筆正楷)： 學號：

專題生簽名(親筆正楷)： 學號：

中華民國 年 月 日

誌　謝

首先感謝我們專題的指導老師林正平老師，謝謝他對我們的悉心指導。

正平老師平常有自己的課要教導其他學生，但在我們製作專題的每個階段，從專題題目的發想，專題題目的確定，中期專題的修改，後期專題格式調整等都給予了許多建議與指導，還特地花費自己的空閒時間替我們演練，給我們多次的練習減少上台失誤的可能性。在此謹向正平老師致以誠摯的謝意和崇高的敬意。同時，感謝在整個專題的製作過程中一起努力的組員們，從決定題目、找資料、擬定大綱、app 的製作、內容修改、美編設計等，大家分工合作各自完成自己的部分，製作過程中沒有爭吵、沒有誰做的多或少，我們接受他人的意見，一起討論一起修改，很有默契的完成了這次的專題。真的是台上一分鐘台下十年功，我們花費了許多的時間，只為了這場六分鐘的專題發表，而專題成果展在組員、同學互相鼓勵之下與老師、評審給予建議之下圓滿、順利的完成。這次的專題發表讓我再次了解到過程的重要性勝於結果，雖然我們沒有做到完美無缺，但至少我們盡心盡力沒有遺憾！再次真誠地向幫助過我們的老師和同學表示感謝！

摘 要

交通繁榮熱鬧的台北，便利的捷運是台北的一大特色，吸引了許多觀光客的前來，至今為止許多觀光客在事先規劃路線上大部分會選擇捷運當作交通工具，但是有些捷運站的出口很多，如果不是很了解的人會很容易迷路，浪費了許多時間在尋找出口，也使得觀光客因為迷路延誤了他們的行程，導致在規畫行程時，在交通方面需要安排更多的時間，因此我們有了想要改善這個問題的動機與想法。

有鑑於此，我們利用 React Native 工具製作了一個使用捷運出口的 APP，APP 的功能簡單分為兩個部分，一是使用者只須下載捷出高手 APP 輸入起點與終點，即可得到路線資訊，利用這個 APP 能讓使用者簡單、方便，並輕鬆得到相關建議，讓找路的程序更為快速，對於未來越來越繁雜的捷運出口路線，能有效率的去尋找最佳出口。二是根據使用者的需求，選擇想前往的路線、車站、行駛方向、出口，便能快速取得完善的出口資訊。

目前本系統僅是雛形系統，未來若是以相同作法，優化介面、加入更多方便的功能，可針對使用者的多樣需求……，提供搭乘電梯或手扶梯的選項，從車廂出來方便找到電梯的位置。最佳路線搜尋功能即時找到最短路程、可應使用者要求更改路線等等，讓本系統能完善發揮資訊的功能。

關鍵字：捷運出口、React Native、APP。

目　錄

圖　目　錄

表　目　錄

第一章 緒論

第一節　系統發展背景

在現今社會中智慧型手機普遍化之下，為了讓生活更便利， 開發各種 APP 應用程式，不僅服務越來越多元化，也廣泛應用在各個領域中，例如：各種訂位、休閒娛樂、搜尋引擎、大眾交通、衛星定位等等……這些科技的進步讓我們的生活越來越便利，因此讓我們產生了一些想法，如何運用 APP 讓複雜的捷運出口簡單明瞭。

近幾年隨著智能化、信息化時代來臨，交通相關的 APP 越來越多樣、進步。日常生活中上班上課、下班下課，捷運是人們不可或缺的交通工具，那如何讓我們搭捷運時更加便利與省時呢？那就是捷運出口查詢，使用者只需拿出手機下載此 APP，輸入行程規劃的地址便能快速地查詢離目的地最近的捷運出口，也可以直接運用出口查詢，選擇您想要的路線、車站、行駛方向、出口，便能快速取得出口資訊。這種結合「高效、便捷、省時、創意」的 APP，不但操作簡單好上手，也省去找尋車廂、車門與出口的繁瑣時間。

此 APP 應用程式適用於每個人，其較針對學生、上班族、觀光客做開發。使早晨趕上課、上班的學生和上班族能節省瑣碎的時間，也使觀光客有完善的交通轉乘資訊，能快速的到達出口。

第二節　問題與動機

隨著科技的發展路線也越來越多樣化可以去到的地方也越來越多，但是隨之而來的問題是怎麼到達目的地、怎麼走才會是最方便的或是怎麼走才會是最快的。但是每次去到了台北車站之類錯綜複雜的地方就會讓人眼花撩亂，覺得無法辨別自己要去的地方是要走哪裡才會抵達，當嘗試用手機查詢也會因為路線太複雜再比對上也會花費很多的時間，甚至還有可能會依然看不懂，所以很多觀光客往往都會浪費許多時間在找路上面，就連長時間居住在台北地區的本地人也都有可能會迷路不知道怎麼走，更何況是剛到沒有多久的觀光客想要不迷路都難。

由於現代科技發展快速，人人一支手機隨處可見，從手機 APP 開始越來越多後，引發我們聯想製作 APP 的路線系統，可以讓想要查詢路線的人只要拿出手機打開 APP 查詢就可以知道要怎麼走、走哪一條路，能夠縮短查詢路線的時間，並且查詢時會給出詳細的路線，避免大量的查詢比對地圖路線等等。而現代人追求方便，為了大眾的便利與時間，不必浪費大量時間進行網路查詢，也利用此系統可以減少大量不必要的時間浪費，避免長久以來迷路的嚴重問題，同時給使用者帶來不一樣的交通新體驗。

第三節　目的

我們利用 React Native 製作台北捷運轉程系統，使用 APP 輸入起點及目的地，將規劃從捷運車廂到出口及目的地最短的路程，避免下車後找出口花費的時間，節省時間，讓使用者可以順利到達目的地，減少轉乘時間及迷路的可能性，無論是上班族、學生、觀光客、趕時間的人，只要運用 APP 都能讓轉程變得簡單、明瞭並快速地到達目的地。

圖 1 APP 首頁圖

第四節　　建構方法與流程

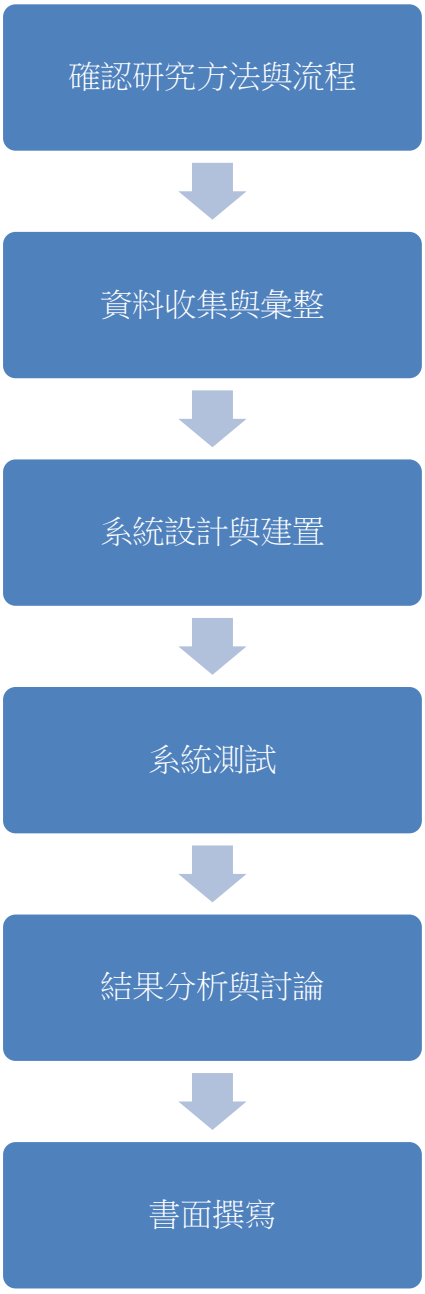


圖 2 建構流程圖

第五節　預期效益

一、透過 APP 快速查詢離目的地最近之車廂及出口，減短找路時間。

二、擁有交通轉乘完善資訊。

三、使用者可直接透過捷出高手 APP 搜尋路線，快速找到離目的地最近出口的車廂，利用手機 APP 輸入想要前往的地點，便能得到清楚明確的路線建議，節省到處問路的時間，讓使用者可順利抵達出口。

四、響應現代人省時省力的精神，減少多餘的時間浪費。

第二章 開發工具

第一節 APP 開發工具

表 1 APP 開發工具表

工具名稱	工具類型
React Native	開發框架
Visual Studio Code	編輯器
Google Map API	地圖服務
Android	作業系統

第二節 APP 開發工具簡介

一、 Android 簡介

Android 是 Google 於 2007 年 11 月推出的開放性嵌入式作業系統平台，它提供了一個包含作業系統、中間層及應用程式的軟體堆疊架構。透過不同作業系統業者所提供的軟體開發工具包（Software Development Kit，SDK）以及高階的程式語言（如 Java 或 Objective-C），讓應用程式開發者可以完全不需要瞭解手機硬體的內部構造與韌體語言便能自行發揮創意巧思去設計開發可安裝於智慧型手機之中的應用程式。

二、 React Native 簡介

2015 年 3 月份的時候，Facebook 開放了 React Native 的原始碼，這是一個能夠讓你利用 JavaScript 建立原生 iOS 及 Android APP 的框架。

三、 Visual Studio Code 簡介

Visual Studio Code 是一個由微軟開發，同時支援 Windows、Linux 和 macOS 等操作系統且開放原始碼的程式碼編輯器，它支援測試，並內建了 Git 版本控制功能，同時也具有開發環境功能，例如程式碼補全（類似於 IntelliSense）、程式碼片段和程式碼重構等，該編輯器支援用戶個性化組態，例如改變主題顏色、鍵盤捷徑等各種屬性和參數，同時還在編輯器中內建了擴充程式管理的功能。

四、 Google Map API 簡介

API 是 Application Programming Interface 的縮寫，亦即應用程式介面。眾所皆知 Google 提供了相當豐富的服務，像是 Google Map、YouTube、Google 文件等，但這些服務通常需要連去 Google 網站才能使用，為此 Google 就提供了以 JavaScript 為基礎的"介面"，讓我們能在自己的網頁上面，利用 JavaScript 語法來呼叫 Google 提供的服務。

第三章　系統功能與特色

第一節　　系統功能

圖 3 系統架構圖

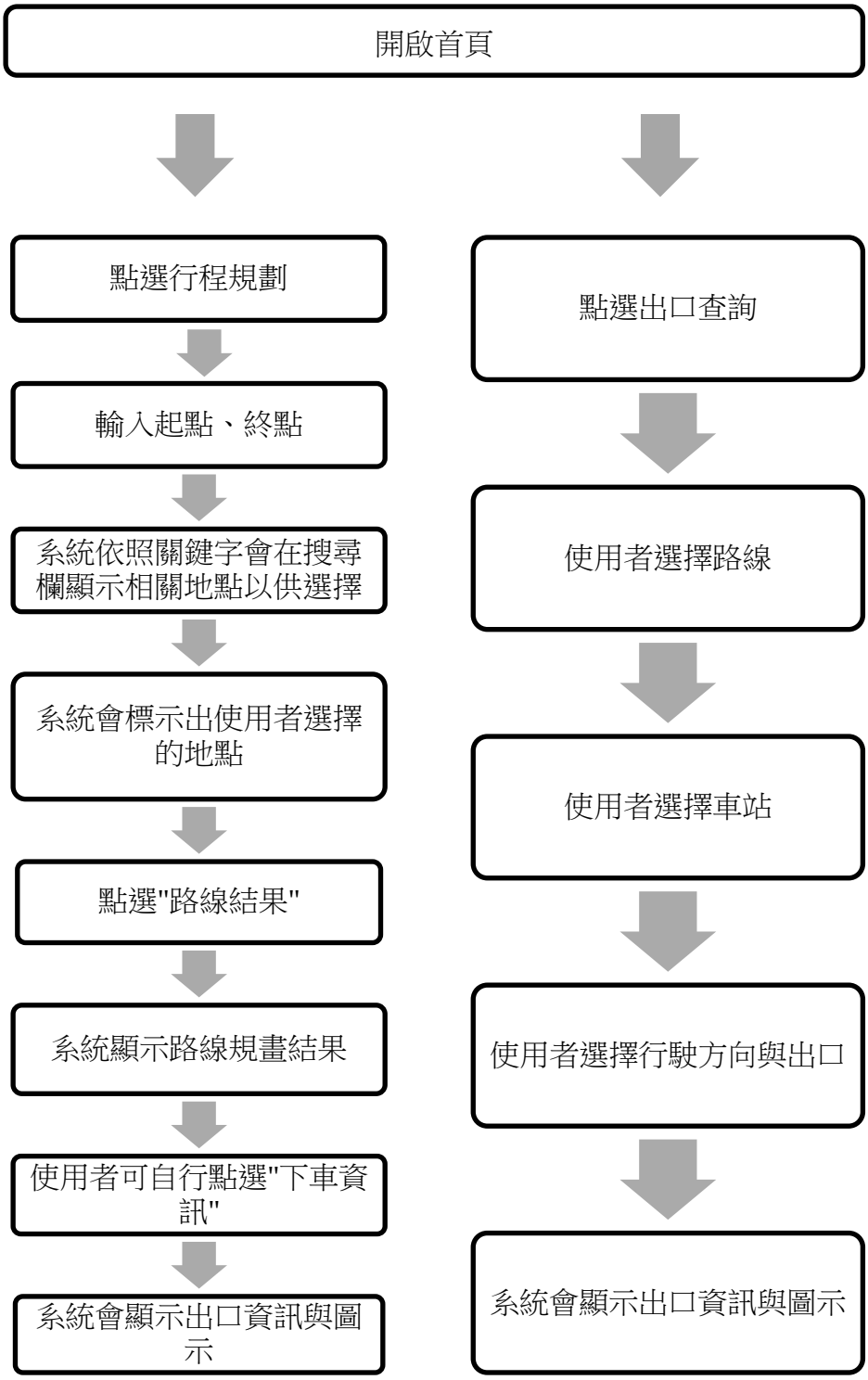


圖 4 系統功能使用說明圖

第二節　系統特色

(一) APP 特色

1. 使用者可以根據自己想要去的地方選擇目的地。
2. APP 會根據使用者選擇的目的地計算出搭乘時間。
3. APP 會給出從哪個車廂上下車比較方便。
4. 介面簡單好操作，使用者能夠即時上手。
5. 根據輸入的內容去尋找距離目的地最近的出入口。
6. 搜尋記錄不會儲存下來也不會公開。

(二) 行程規劃

1. 使用者即使不知道在哪一站下車，只要照著查詢路線走，就可以輕鬆到達目的地。
2. 以最短的距離到目的地，讓使用者節省時間、不迷路。

(三) 出口查詢

1. 使用者可以選擇搭乘的路線、方向及出口，清楚又明瞭。
2. 以簡單的文字搭配清楚的圖像讓使用者得知從哪個車廂上/下車離指定的出口最接近。

第四章 使用對象與環境

搭捷運下車時要在哪個車廂離出口比較近一直以來都是乘客難以預料的事情，往往都要多走很多路才能到達欲前往的出口，越多出口的捷運站，越是急需改善這方面的問題，遇到上下班的人群壅塞時段，如果沒辦法準確猜測到離出口最近的車廂，就需要花多餘的時間人擠人再走一段路，這也是為什麼大多數的乘客在到站後都快走甚至是用跑的搶著下車，雖然省去了麻煩卻也造成了安全問題，搶快時可能會跌倒受傷，而我們的 APP 程式雖然說不上可以完全解決，但已經能夠暫時緩解這方面的問題。本 APP 能夠事先告知乘客在哪個車廂下車離欲前往的出口較近。除此之外，我們的 APP 還有路程規劃功能，透過路程規劃我們可以直接查詢該從哪個出口出去，不用再花時間上網查。透過捷出高手 APP，讓乘客可以方便的用智慧型手機查詢車廂出口以及路程規劃，不需要再花費多餘的時間及體力，這也體現了現代科技生活中追求極簡的生活理念。

以後如果有任何時候需要查詢路或者是出口，皆可以使用這一個路線 APP，簡化了查詢路線作業上瑣碎的流程與節省大量的時間成本，不但顧及使用者的心情，也能立即掌握現在要到達目的地最近的出口方向。在 APP 上，使用者能夠看到起始點、目的地，對於使用者來說，這些內容已經有符合大部分的路線需求，能夠隨時都能透過 APP 檢視，相當方便不需要像過去一樣利用地圖或是搜尋引擎等等的方式去查詢路線，且透過系統快速掌握、查詢路線，到達目的地之後也不用煩惱使用者自己的搜尋紀錄會不會被記錄下來或是公開等等的問題，以 APP 為主要實踐方式，以降低使用者浪費的時間，同時也期望使用 APP 後，可以減少使用者查詢路線的辛苦，提升使用者的交通品質。

第五章 系統實作

第一節　行程規劃

圖 5 首頁圖

此 APP 有兩大主要頁面，分別為行程規劃與出口查詢，圖 為 APP 首頁畫面的編排。

```
import React, { FC, ReactElement } from 'react';
import {
  View,
  Text,
  StyleSheet,
  GestureResponderEvent,
  ScrollView
} from 'react-native';
import { theme } from '../../theme/theme';
import { Button } from 'react-native-elements';
import { RouteComponentProps } from 'react-router';
import Icon from 'react-native-vector-icons/FontAwesome';
```

圖 6 首頁程式宣告圖

```
const NavigatorButton: FC<{
  title: string;
  onPress?: (event: GestureResponderEvent) => void;
  icon: ReactElement;
}> = ({ title, onPress, icon }) => (
  <View style={styles.navigatorButtonContainer}>
    <Button
      title={title}
      buttonStyle={styles.navigatorButton}
      titleStyle={styles.navigatorButtonTitle}
      raised
      onPress={onPress}
      icon={icon}
    ></Button>
  </View>
);
```

圖 7 導覽按鈕元件

```tsx
export const HomePage: FC<RouteComponentProps> = ({ history }) => {
  return (
    <View style={styles.pageContainer}>
      <View style={styles.header}>
        <Text style={styles.headerTitle}>捷運出口</Text>
      </View>
      <ScrollView contentContainerStyle={styles.scrollView}>
        <NavigatorButton
          title="行程規劃"
          onPress={() => history.push('/route-plan')}
          icon={<Icon name="edit" size={90} color="#fff" />}
        />
        <NavigatorButton
          title="出口查詢"
          onPress={() => history.push('/select-line-page')}
          icon={<Icon name="search" size={90} color="#fff" />}
        />
      </ScrollView>
    </View>
  );
};
```

圖 8 首頁程式元件

圖 8 為 APP 首頁的程式元件，包含了行程規劃與出口查詢的按鈕、按鈕的圖示、按鈕按了之後會切換到對應的頁面等……。

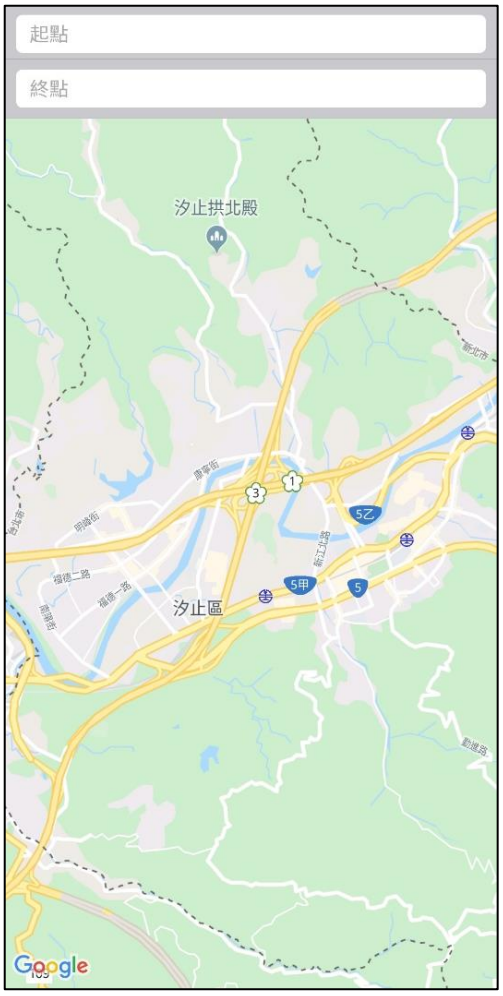


圖 9 行程規劃頁

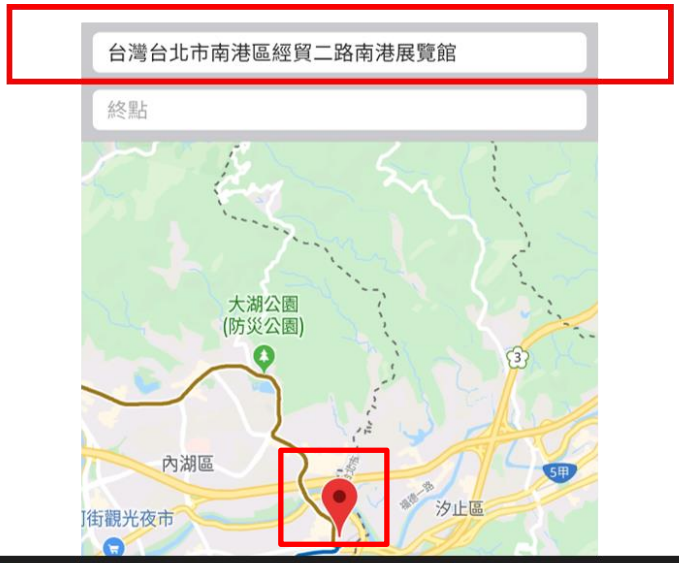
圖 9 為行程規劃頁，使用了 Google 的 Map API、自動建議關鍵字與其規劃路線功能，讓使用者輸入起點終點後，可點選路線結果跟下車資訊得知規劃的路線與捷運車廂出口。

```
import React, {
  FC,
  useState,
  ReactElement,
  useRef,
  useEffect,
  Component
} from 'react';
import MapView, { PROVIDER_GOOGLE, Callout, Marker } from 'react-native-maps';
import { StyleSheet, View, Text, ScrollView } from 'react-native';
// @ts-ignore
import { GooglePlacesAutocomplete } from 'react-native-google-places-autocomplete';
import MapViewDirections from 'react-native-maps-directions';
import { Button } from 'react-native-elements';
import axios from 'axios';
import { Route, RouteComponentProps } from 'react-router';
// @ts-ignore
import HTML from 'react-native-render-html';
import { useDispatch, useSelector } from 'react-redux';
import { AppDispathTypes, AppState } from '../../store';
import {
  setCurrentDirection,
  setCurrentStationId,
  setCurrentLinAlias,
  setCurrentStationName
} from '../../store/serchInfoState/actions';
import { stationList } from '../../utils/exitData';
import * as H from 'history';
import { Step, MapStepInfo } from '../../types';
import {
  setMapStepState,
  setCurrentLocation,
  setEndLocation,
  setOriginMaker,
  setDestinationMaker,
  setStartSearchText,
  setEndSearchText,
  resetSearchLocation
} from '../../store/routePlanInfoState/actions';
import { stationList2 } from '../../utils/exitData2';
import { SelectExitPage } from '../select-exit-page/select-exit-page';
import { ExitInfoPage } from '../exit-info-page/exit-info-page';
```

圖 10 行程規劃程式宣告圖

圖 11 行程規劃頁之起點欄位程式碼圖-1

當在起點文字欄輸入大於等於 2 個字元時，系統就會自動建議關鍵字。



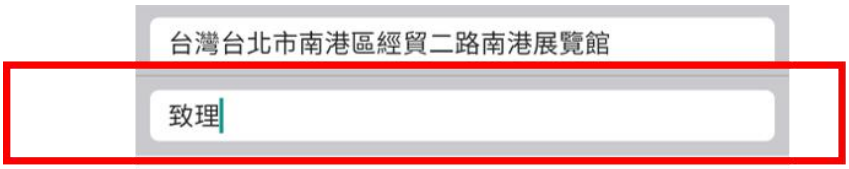

```
onPress={(data: any, details: any) => {
  // 設定起點的搜尋文字，讓頁面切換回來時能設定回搜尋的入框
  dispatch(setStartSearchText(data.description));
  // 設定起點位置
  dispatch(
    setCurrentLocation({
      latitude: details.geometry.location.lat,
      longitude: details.geometry.location.lng,
      latitudeDelta: 0.0922,
      longitudeDelta: 0.0421
    })
  );
  // 設定起點圖釘位置
  dispatch(
    setOriginMaker({
      coordinate: {
        latitude: details.geometry.location.lat,
        longitude: details.geometry.location.lng
      },
      title: details.formatted_address,
      description: details.formatted_address
    })
  );
}}
```

圖 12 行程規劃頁之起點欄位程式碼圖-2

當點選系統建議的地點時，系統會在選取之地點加上圖釘。

圖 13 行程規劃頁之起點欄位程式碼圖-3

起點欄位預設值為空白，當執行 Google Map API 時，需要提供申請之 API Key，且將查詢結果語言設定為中文。



台灣新北市板橋區文化路一段致理科技大學

台灣新北市板橋區致理科大

台灣新北市板橋區文化路一段致理科技大學圖書館

台灣新北市板橋區文化路一段 致理專業影印

台灣新北市板橋區介壽街致理幼兒園

powered by Google

```
{/* 搜尋終點的輸入框 */}
<GooglePlacesAutocomplete
  ref={endSearchInputRef}
  placeholder="終點"
  minLength={2} // minimum length of text to search
  autoFocus={false}
  fetchDetails={true}
  textInputProps={{
    onFocus: () => {
      setShowResultBtn(false);
    },
    onBlur: () => {
      setShowResultBtn(true);
    },
    onChange: () => {
      dispatch(setEndLocation(null));
      dispatch(setDestinationMaker(null));
    }
  }}
```

圖 14 行程規劃頁之終點欄位程式碼圖-1

當在終點文字欄輸入大於等於 2 個字元時，系統就會自動建議關鍵字。


```
onPress={(data: any, details: any) => {
  // 設定終點的搜尋文字，讓頁面切換回來時能設定回搜尋的入框
  dispatch(setEndSearchText(data.description));
  // 設定終點位置
  dispatch(
    setEndLocation({
      latitude: details.geometry.location.lat,
      longitude: details.geometry.location.lng,
      latitudeDelta: 0.0922,
      longitudeDelta: 0.0421
    })
  );
  // 設定終點圖釘位置
  dispatch(
    setDestinationMaker({
      coordinate: {
        latitude: details.geometry.location.lat,
        longitude: details.geometry.location.lng
      },
      title: details.formatted_address,
      description: details.formatted_address
    })
  );
}}
```

圖 15 行程規劃頁之終點欄位程式碼圖-2

當點選系統建議的地點時，系統會在選取之地點加上圖釘。

圖 16 行程規劃頁之終點欄位程式碼圖-3

終點欄位預設值為空白，當執行 Google Map API 時，需要提供申請之 API Key，且將查詢結果語言設定為中文。

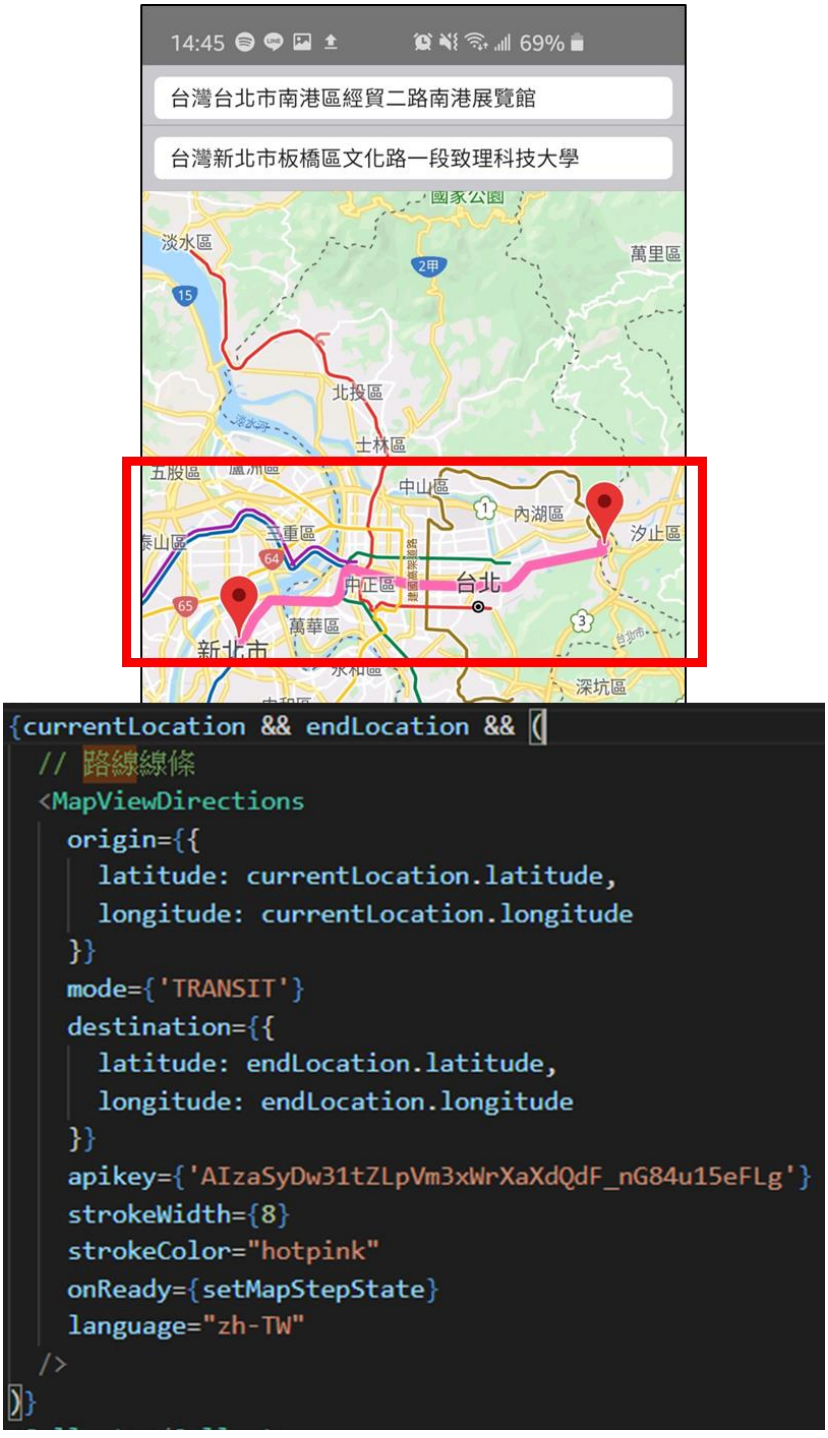


圖 17 行程規劃頁之終點欄位程式碼圖-4

此處為繪製路線線條的程式碼，利用起點與終點的經緯度搭配 API 且限定使用大眾運輸所繪製出來的線條。

圖 18 行程規劃頁之終點欄位程式碼圖-5

當起點與終點皆設定好之後，下方的「路線結果」按鈕會自動提示出來讓使用者點選，點選後即可得到路線建議與捷運車廂出口。

圖 19 路線結果頁

路線結果頁面可以得知路線的規畫結果，配合箭頭處的 1 出口提示，點擊方框處可以得知要在哪個車廂與車門下車會離 1 號出口最接近。

```
/**
 * 利用 google map api 取得路線階段
 */
async function getRouteSteps() {
  if (currentLocation && endLocation) {
    try {
      /** 2019-11-28 12:00:00 timestamp */
      const commonDepartureTime = 1574222400;
      // 發出請求
      const { data } = await axios.get<MapStepInfo>(
        `https://maps.googleapis.com/maps/api/directions/json?origin=${currentLocation.latitude},${currentLocation.longitude}&destina
        tion=${endLocation.latitude},${endLocation.longitude}&key=AIzaSyDw31tZLpVm3xWrXaXdQdF_nG84u15eFLg&language=zh-TW&mode=transit`
      );
      dispatch(setMapStepState(data));
    } catch (error) {
      dispatch(setMapStepState(error));
    }
    history.push('/route-plan/result');
  }
}
```

圖 20 路線結果頁之路線規劃程式碼圖-1

```
/**
 * 渲染路線階段
 */
function RenderSteps({
  step,
  history
}: {
  step: Step;
  history: H.History;
}): ReactElement {
  const dispatch = useDispatch<AppDispathTypes>();
  /**
   * 檢查目前的路線，回傳路線代號
   * @param {string} to 前往路線方向
   */
  const checkCurrentLine = (to: string) => {
    if (to.includes('新店') || to.includes('松山')) {
      return 'G';
    } else if (
      to.includes('南港展覽館') ||
      to.includes('亞東醫院') ||
      to.includes('頂埔')
    ) {
      return 'BL';
    }
    return '';
  };
  /**
   * 檢查方向，回傳方向
   * @param to 前往路線方向
   */
  const checkDirection = (to: string) => {
    if (to.includes('新店') || to.includes('亞東醫院') || to.includes('頂埔')) {
      return 'direction';
    } else if (to.includes('南港展覽館') || to.includes('松山')) {
      return 'directionR';
    }
    return 'direction';
  };
```

圖 21 路線結果頁之路線規劃程式碼圖-2


```
return (
  <>
    {step.html_instructions && !step.steps && (
      <View style={{ flexDirection: 'row', marginBottom: 10 }}>
        <View
          style={{
            flexBasis: '80%',
            flexDirection: 'row',
            padding: 10,
            backgroundColor: '#e0e0e0',
            shadowColor: '#000',
            shadowOffset: {
              width: 0,
              height: 4
            },
            shadowOpacity: 0.32,
            shadowRadius: 5.46,
            elevation: 1
          }}
        >
          <View style={{ flexGrow: 1 }}>
            <HTML
              html={step.html_instructions}
              key={step.html_instructions}
            />
            {step.transit_details && step.transit_details.arrival_stop && (
              <View>
                <Text>{`於 ${step.transit_details.arrival_stop.name} 下車`}</Text>
              </View>
            )}
```

圖 22 路線結果頁之路線規劃程式碼圖-3

此處把從 Google Map API 得到的資料抓取需要的部分，並條列式顯示出來。


```
{step.html_instructions.includes('開往') &&
  step.html_instructions.includes('捷運') &&
  step.transit_details &&
  step.transit_details.arrival_stop && (
    <Button
      title={'下車資訊'}
      buttonStyle={{ flexBasis: 50 }}
      titleStyle={{ fontSize: 16 }}
      raised
      onPress={() => {
        const direction = checkDirection(step.html_instructions);
        // 下車車站
        const arrivalStop = step.transit_details!.arrival_stop!.name.includes(
          '站'
        ) /* 去除台北車站以外的站名裡詞尾的 '站' 字 */
          ? step.transit_details!.arrival_stop!.name === '台北車站'
            ? step.transit_details!.arrival_stop!.name
            : step.transit_details!.arrival_stop!.name.slice(0, -1)
          : step.transit_details!.arrival_stop!.name;
        /* 取得對應路線的車站列表 */
        const currentStationList =
          checkCurrentLine(step.html_instructions) === 'G'
            ? stationList
            : checkCurrentLine(step.html_instructions) === 'BL'
            ? stationList2
            : [];
        const currentStation = currentStationList.find(
          station => station.stationName === arrivalStop
        );
```

圖 23 路線結果頁之路線規劃程式碼圖-4

圖 24 路線結果頁之路線規劃程式碼圖-5


```
            <View
              style={{
                flexBasis: '20%',
                paddingLeft: 10,
                flexDirection: 'row',
                alignItems: 'center',
                height: '100%'
              }}
            >
              <Text>{step.duration.text}</Text>
            </View>
          </View>
        )}
        {step.steps &&
          step.steps.map((step, i) => (
            // 如果路線階段有巢狀階段，則再繼續渲染巢狀階段
            <RenderSteps key={i} step={step} history={history}></RenderSteps>
          ))}
      </>
    );
}
```

圖 25 路線結果頁之路線規劃程式碼圖-6

圖 26 下車資訊頁-1

使用者可配合圖 19 箭頭提示之出口，自行點選出口。

配合圖 23 的程式碼，判斷從 API 取得的開往「亞東醫院站」為板南線的順行路線，等於選擇行駛方向頁面中的「往頂埔」，由於 Google Map 中的各站名稱有時候會有含有「站」這個字（例如：松山站、西門站），但我們設定的資料中，除了台北車站外，其他都是只有單純的站名（例如：松山、西門），所以需要去除掉台北車站以外的「站」字，以便程式判斷（圖 23），直接點選下車資訊即可帶來選擇新埔站之出口。

圖 27 下車資訊頁-2

使用者即可得到出口資訊與車站剖面圖。

第二節　出口查詢

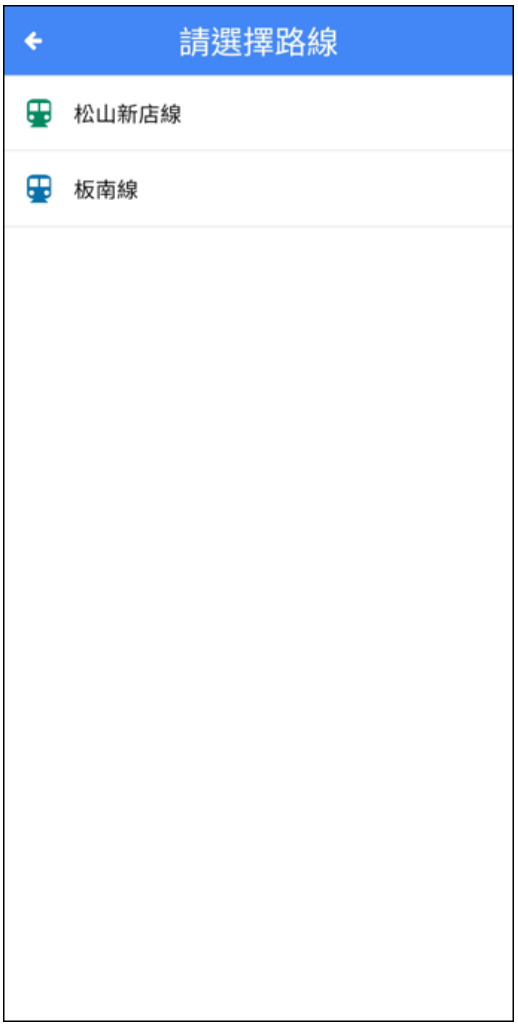


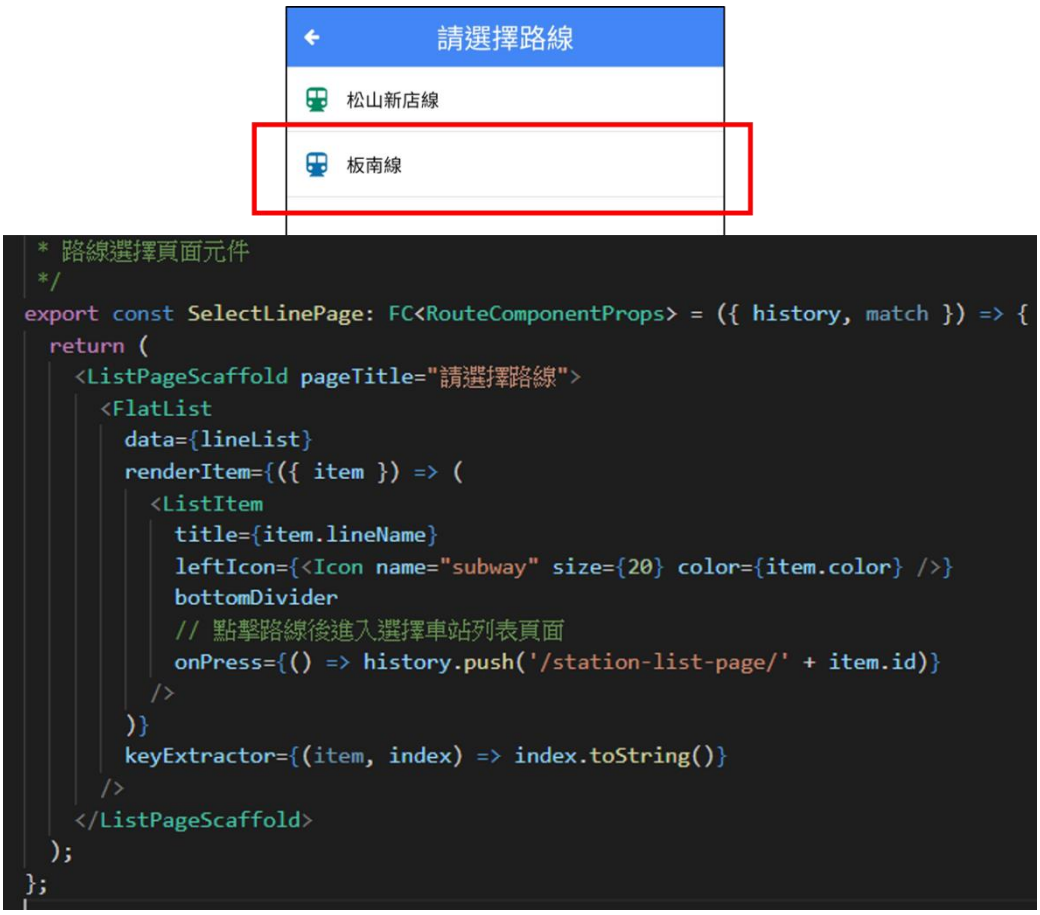
圖 28 出口查詢首頁圖

出口查詢頁面是讓已經知道要去哪一站、哪一個出口的使用者所使用，只要點選幾個選項即可知道離目的出口最近之捷運車廂。

```
import React, { FC } from 'react';
import { FlatList } from 'react-native';
import { RouteComponentProps } from 'react-router';
import Icon from 'react-native-vector-icons/FontAwesome';
import { ListItem } from 'react-native-elements';
import { ListPageScaffold } from '../list-page-scaffold/list-page-scaffold';
```

圖 29 選擇路線頁面之程式宣告圖

圖 30 選擇路線頁面之路線列表圖




```
 * 路線選擇頁面元件
 */
export const SelectLinePage: FC<RouteComponentProps> = ({ history, match }) => {
  return (
    <ListPageScaffold pageTitle="請選擇路線">
      <FlatList
        data={lineList}
        renderItem={({ item }) => (
          <ListItem
            title={item.lineName}
            leftIcon={<Icon name="subway" size={20} color={item.color} />}
            bottomDivider
            // 點擊路線後進入選擇車站列表頁面
            onPress={() => history.push('/station-list-page/' + item.id)}
          />
        )}
        keyExtractor={(item, index) => index.toString()}
      />
    </ListPageScaffold>
  );
};
```

圖 31 選擇路線頁面之程式元件

點選路線之後會進入到選擇車站的頁面。

圖 32 選擇車站頁面圖

選擇路線後，將進入選擇車站頁面，對應的路線會有相對顏色的標籤。

```
import React, { FC } from 'react';
import { View, FlatList, StyleSheet } from 'react-native';
import { RouteComponentProps } from 'react-router';
import { ListItem } from 'react-native-elements';
import { ListPageScaffold } from '../list-page-scaffold/list-page-scaffold';
import { stationList } from '../../utils/exitData';
import { stationList2 } from '../../utils/exitData2';

import { useDispatch } from 'react-redux';
import { AppDispathTypes } from '../../store';
import {
  setCurrentLinAlias,
  setCurrentStationId,
  setCurrentStationName
} from '../../store/serchInfoState/actions';
```

圖 33 選擇車站頁面之程式宣告圖

圖 34 出口資料圖

此圖是我們將收集到的出口資料整理成陣列，裡面有出口代號、出口名稱、行駛方向、車廂代號、車門代號。

表 2 順行逆行示意圖

	行駛方向（順行）	行駛方向（逆行）
板南線	往頂埔	往南港展覽館
松山新店線	往新店	往松山


```
 * 車站列表頁面元件
 */
export const StationListPage: FC<RouteComponentProps<{ id: string }>> = ({
  match,
  history
}) => {
  const dispatch = useDispatch<AppDispathTypes>();

  return (
    <ListPageScaffold pageTitle="請選擇車站">
      <FlatList
        data={match.params.id === '1' ? stationList : stationList2}
        renderItem={({ item }) => (
          <ListItem
            title={item.stationName}
            leftIcon={
              <View
                style={
                  match.params.id === '1' ? styles.tagLineG : styles.tagLineBL
                }
              ></View>
            }
            bottomDivider
            onPress={() => {
              // 將選擇的車站的資料儲存到中央資料層
              dispatch(setCurrentLinAlias(item.lineAlias));
              dispatch(setCurrentStationId(item.stationId));
              dispatch(setCurrentStationName(item.stationName));
              // 進入選擇方向頁面
              history.push('/select-direction-page');
            }}
          />
        )}
        keyExtractor={(item, index) => index.toString()}
      />
    </ListPageScaffold>
  );
};
```

圖 35 選擇車站頁面之程式元件圖

點選車站之後會進入到選擇行駛方向的頁面。

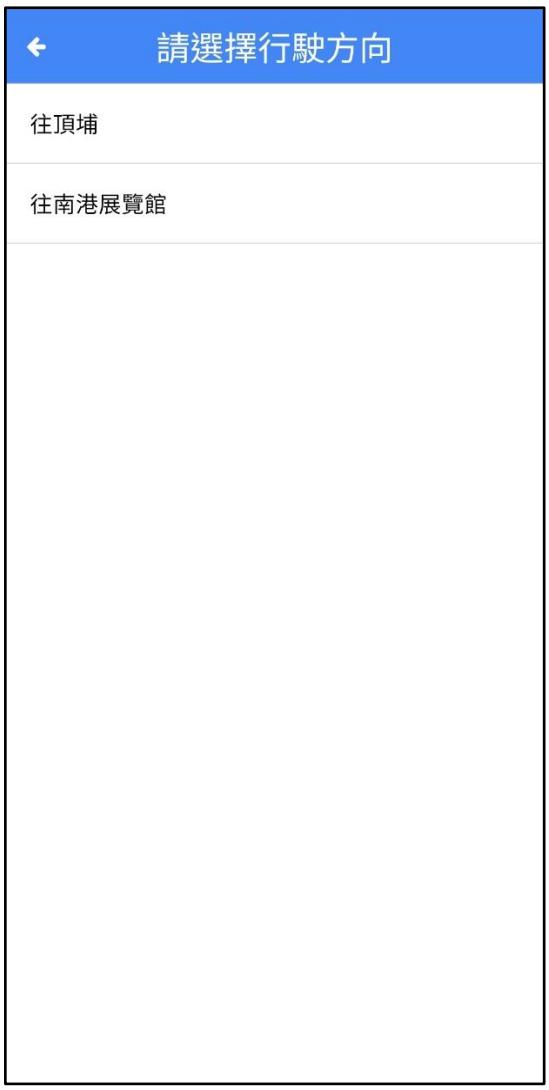


圖 36 選擇行駛方向頁面圖

點選車站之後會進入到選擇行駛方向的頁面。

```
import React, { FC, useMemo } from 'react';
import { View, ScrollView } from 'react-native';
import { RouteComponentProps } from 'react-router';
import { ListItem } from 'react-native-elements';
import { ListPageScaffold } from '../list-page-scaffold/list-page-scaffold';
import { stationList } from '../../utils/exitData';
import { useSelector, useDispatch } from 'react-redux';
import { AppState, AppDispathTypes } from '../../store';
import { setCurrentDirection } from '../../store/serchInfoState/actions';
import { stationList2 } from '../../utils/exitData2';
```

圖 37 選擇行駛方向頁面之程式宣告圖

```
/**
 * 方向選擇頁面元件
 */
export const SelectDirectionPage: FC<RouteComponentProps<{
  stationId: string;
}>> = ({ history }) => {
  // 目前車站的編號
  const currentStationId = useSelector(
    (state: AppState) => state.serchInfoState.currentStationId
  );
  // 目前路線代號
  const currentLineAlias = useSelector(
    (state: AppState) => state.serchInfoState.currentLineAlias
  );
  // 發送資料到中央資料層的 function
  const dispatch = useDispatch<AppDispathTypes>();
  // 首站資訊
  const firstStationInfo = useMemo(
    () => (currentLineAlias === 'G' ? stationList[0] : stationList2[0]),
    [currentLineAlias]
  );
  // 底站資訊
  const lastStationInfo = useMemo(
    () =>
      currentLineAlias === 'G'
        ? stationList.slice().reverse()[0]
        : stationList2.slice().reverse()[0],
    [currentLineAlias]
  );
```

圖 38 選擇行駛方向頁面之程式元件圖-1


```
    <ListPageScaffold pageTitle="請選擇行駛方向">
      <View style={{ flexGrow: 1 }}>
        <ScrollView>
          {currentStationId !== lastStationInfo.stationId && (
            <ListItem
              title={`往${firstStationInfo.stationName}`}
              bottomDivider
              onPress={() => {
                dispatch(setCurrentDirection('direction'));
                history.push('/select-exit-page');
              }}
            />
          )}
          {currentStationId !== firstStationInfo.stationId && (
            <ListItem
              title={`往${lastStationInfo.stationName}`}
              bottomDivider
              onPress={() => {
                dispatch(setCurrentDirection('directionR'));
                history.push('/select-exit-page');
              }}
            />
          )}
        </ScrollView>
      </View>
    </ListPageScaffold>
  );
};
```

圖 39 選擇行駛方向頁面之程式元件圖-2

點選行駛方向之後會進入到選擇出口的頁面。

圖 40 選擇出口頁面圖

選擇行駛方向後會進入到選擇出口的頁面。

```
import React, { FC, useMemo } from 'react';
import { ListPageScaffold } from '../list-page-scaffold/list-page-scaffold';
import { Text, FlatList, View } from 'react-native';
import { stationList } from '../../utils/exitData';
import { useSelector, useDispatch } from 'react-redux';
import { AppState, AppDispathTypes } from '../../store';
import { ListItem } from 'react-native-elements';
import { setCurrentExitInfo } from '../../store/serchInfoState/actions';
import { RouteComponentProps } from 'react-router';
import { stationList2 } from '../../utils/exitData2';
```

圖 41 選擇出口頁面之程式宣告圖

```
interface IProps {
  basePath?: string;
}
/**
 * 出口選擇頁面元件
 */
export const SelectExitPage: FC<IProps & RouteComponentProps> = ({
  basePath = '',
  history
}) => {
  // 目前車站的編號
  const currentStationId = useSelector(
    (state: AppState) => state.serchInfoState.currentStationId
  );
  // 目前路線代號
  const currentLineAlias = useSelector(
    (state: AppState) => state.serchInfoState.currentLineAlias
  );
  // 發送資料到中央資料層的 function
  const dispatch = useDispatch<AppDispathTypes>();
  // 目前車站的出口列表，如果搜尋不到車站則回傳空陣列
  const currentStationExits = useMemo(() => {
    const searchStationList =
      currentLineAlias === 'G' ? stationList : stationList2;
    const currentStation = searchStationList.find(
      station => station.stationId === currentStationId
    );

    if (currentStation) {
      return currentStation.exit;
    }

    return [];
  }, [currentStationId, currentLineAlias]);
```

圖 42 選擇出口頁面之程式元件圖-1

圖 43 選擇出口頁面之程式元件圖-2

點選出口之後會進入到出口資訊圖的頁面。

圖 44 出口資訊圖

選擇出口後會進入到出口資訊圖的頁面。

```
import React, { FC, useMemo } from 'react';
import { ListPageScaffold } from '../list-page-scaffold/list-page-scaffold';
import {
  View,
  Text,
  StyleSheet,
  ScrollView,
  Image,
  Dimensions
} from 'react-native';
import { useSelector } from 'react-redux';
import { AppState } from '../../store';
import { stationList } from '../../utils/exitData';
import { ListItem } from 'react-native-elements';
import { stationList2 } from '../../utils/exitData2';
import { imageMap } from '../../utils/imageMap';
```

圖 45 出口資訊圖頁面之程式宣告圖

```
/**
 * 出口資訊頁面元件
 */
export const ExitInfoPage: FC = () => {
  // 目前車站的編號
  const currentStationId = useSelector(
    (state: AppState) => state.serchInfoState.currentStationId
  );
  // 目前站名
  const currentStationName = useSelector(
    (state: AppState) => state.serchInfoState.currentStationName
  );
  // 目前的出口資訊
  const currentExitInfo = useSelector(
    (state: AppState) => state.serchInfoState.currentExitInfo
  );
  // 目前方向（順逆）
  const currentDirection = useSelector(
    (state: AppState) => state.serchInfoState.currentDirection
  );
  // 目前路線代號
  const currentLineAlias = useSelector(
    (state: AppState) => state.serchInfoState.currentLineAlias
  );
  // 路線上的首站
  const firstStationInfo = useMemo(
    () => (currentLineAlias === 'G' ? stationList[0] : stationList2[0]),
    [currentLineAlias]
  );
  // 路線上的底站
  const lastStationInfo = useMemo(
    () =>
      currentLineAlias === 'G'
        ? stationList.slice().reverse()[0]
        : stationList2.slice().reverse()[0],
    [currentLineAlias]
  );
```

圖 46 出口資訊圖頁面之程式元件圖-1

```
// 目前前往方向（例:往 xxx 站）
const currentDirectionName = useMemo(
  () =>
    currentDirection === 'direction'
      ? firstStationInfo.stationName
      : lastStationInfo.stationName,
  [currentDirection]
);
// 目前前往方向出口的資訊
const currentExitWithDirectionInfo = useMemo(() => {
  if (currentExitInfo) {
    return currentDirection === 'direction'
      ? currentExitInfo.direction
      : currentExitInfo.directionR;
  }
}, [currentExitInfo, currentDirection]);

/**
 * 取得站內剖面圖
 */
function getImage() {
  return imageMap[currentLineAlias][currentStationId!];
}
```

圖 47 出口資訊圖頁面之程式元件圖-2


```
return (
  <ListPageScaffold pageTitle="出口資訊">
    {!!currentExitInfo && (
      <>
        <View style={styles.exitInfoTipContainer}>
          <Text style={styles.exitInfoTip}>
            <Text>預先站在往</Text>
            <Text style={styles.fontBold}>{currentDirectionName}</Text>
            <Text>月台的</Text>
            <Text style={styles.fontBold}>
              {currentExitWithDirectionInfo &&
                currentExitWithDirectionInfo.carOrder}
            </Text>
            <Text>號車廂，車頭數來第</Text>
            <Text style={styles.fontBold}>
              {currentExitWithDirectionInfo &&
                currentExitWithDirectionInfo.doorId}
            </Text>
            <Text>道門，就能快速抵達</Text>
            <Text style={styles.fontBold}>{currentStationName}</Text>
            <Text>站</Text>
            <Text style={styles.fontBold}>{currentExitInfo.exitId}</Text>
            <Text>號出口</Text>
          </Text>
        </View>
```

圖 48 出口資訊圖頁面之程式元件圖-3

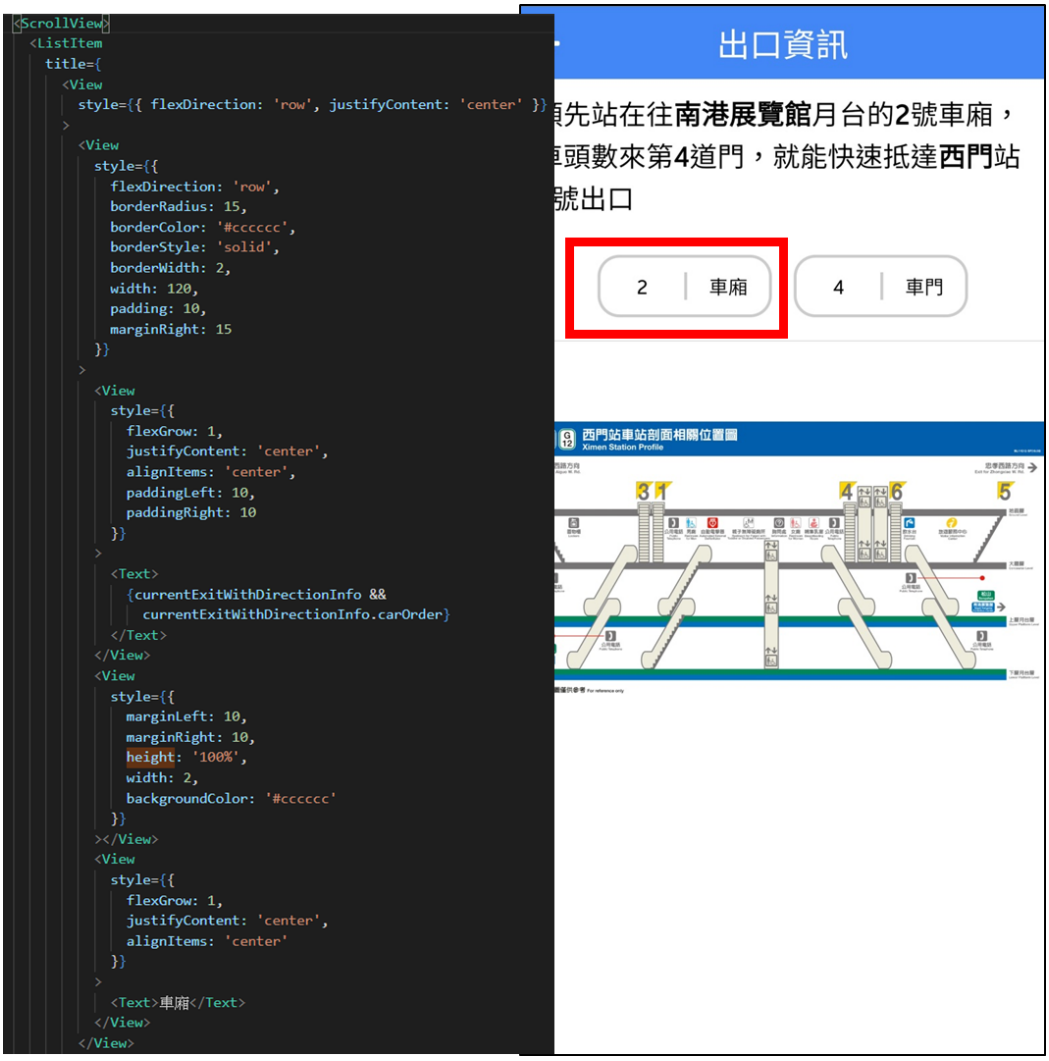
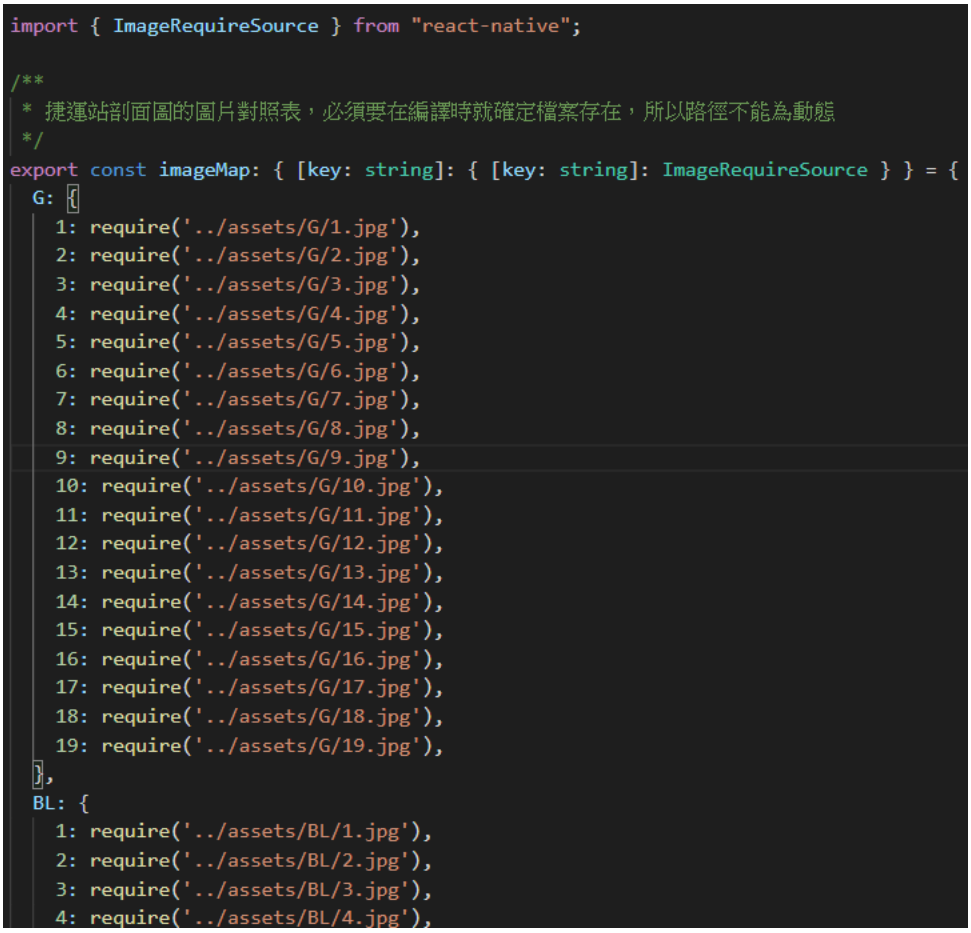


圖 49 出口資訊圖頁面之程式元件圖-4

圖 50 出口資訊圖頁面之程式元件圖-5


```
            <Image source={getImage()} resizeMode="center" style={styles.img} />
          </ScrollView>
        </>
      )}
    </ListPageScaffold>
  );
};
```

圖 51 出口資訊圖頁面之程式元件圖-6



```
import { ImageRequireSource } from "react-native";

/**
 * 捷運站剖面圖的圖片對照表，必須要在編譯時就確定檔案存在，所以路徑不能為動態
 */
export const imageMap: { [key: string]: { [key: string]: ImageRequireSource } } = {
  G: {
    1: require('../assets/G/1.jpg'),
    2: require('../assets/G/2.jpg'),
    3: require('../assets/G/3.jpg'),
    4: require('../assets/G/4.jpg'),
    5: require('../assets/G/5.jpg'),
    6: require('../assets/G/6.jpg'),
    7: require('../assets/G/7.jpg'),
    8: require('../assets/G/8.jpg'),
    9: require('../assets/G/9.jpg'),
    10: require('../assets/G/10.jpg'),
    11: require('../assets/G/11.jpg'),
    12: require('../assets/G/12.jpg'),
    13: require('../assets/G/13.jpg'),
    14: require('../assets/G/14.jpg'),
    15: require('../assets/G/15.jpg'),
    16: require('../assets/G/16.jpg'),
    17: require('../assets/G/17.jpg'),
    18: require('../assets/G/18.jpg'),
    19: require('../assets/G/19.jpg'),
  },
  BL: {
    1: require('../assets/BL/1.jpg'),
    2: require('../assets/BL/2.jpg'),
    3: require('../assets/BL/3.jpg'),
    4: require('../assets/BL/4.jpg'),
```

圖 52 出口資訊圖頁面之捷運站剖面圖資訊

因為路徑不能為動態，所以資料填入方式為靜態，G（松山新店線）有 19 站，就需要 19 個需求，BL（板南線）有 23 站，也就需要 23 個需求。

第六章 系統畫面

圖 53 APP 首頁圖

捷出高手－北捷出口指引 APP 分為兩大功能：行程規劃、出口查詢。

行程規劃主要是讓使用者輸入起點和終點後，便能得到搭乘路線資訊。出口查詢主要是讓使用者查詢欲前往出口之最近車廂。

圖 54 選擇出口查詢圖

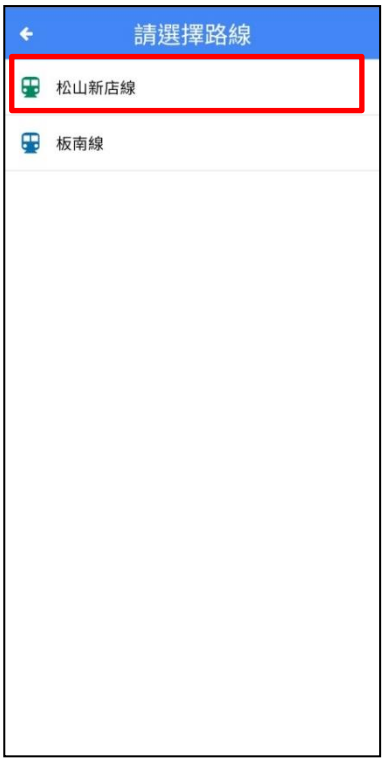


圖 55 選擇路線圖

圖 56 選擇車站圖

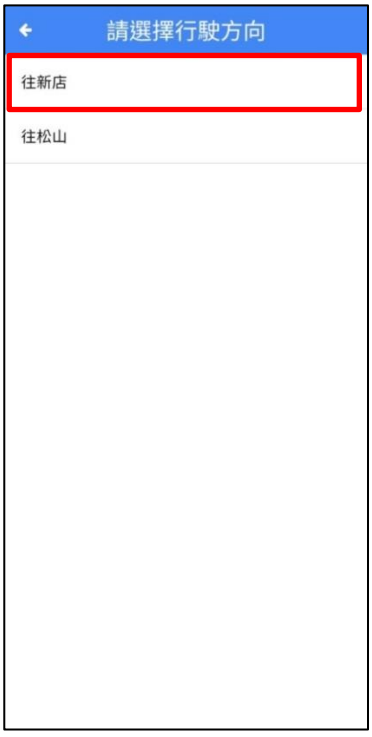


圖 57 選擇行駛方向圖

選擇出口查詢（圖 54）即可選擇路線（圖 55），此以松山新店線為示範。

選擇想到達的車站（圖 56）以西門為例，即可選擇行駛方向（圖 57）以新店為例。

圖 58 選擇出口圖

圖 59 出口資訊圖

選擇想到達的出口（圖 58）以西門町為例，最終結果顯示完善的出口資訊與圖示（圖 58）。

圖 60 選擇行程規劃圖

圖 61 行程規劃頁面圖

圖 62 起點輸入圖

圖 63 起點地標圖

圖 64 終點輸入圖

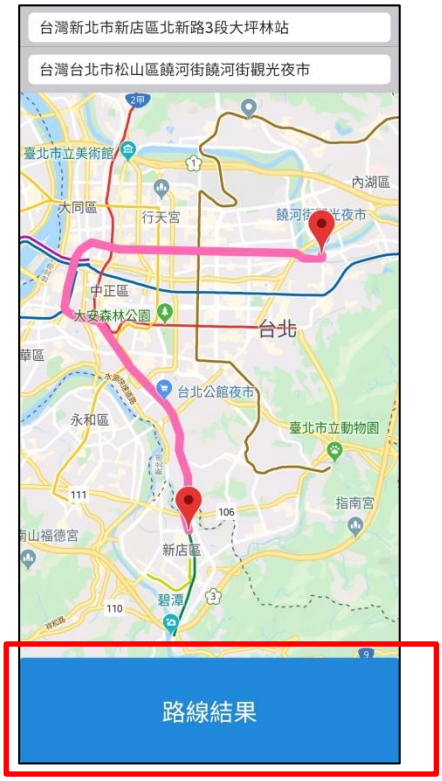


圖 65 行程路線圖

輸入關鍵字會在搜尋欄顯示相關地點以供選擇（圖 62、64），形成路線（圖 65），點選路線結果（圖 65 方框處）。

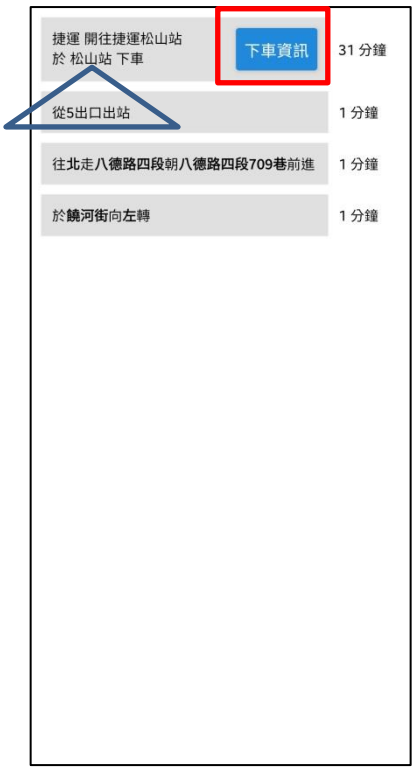


圖 66 路線結果圖

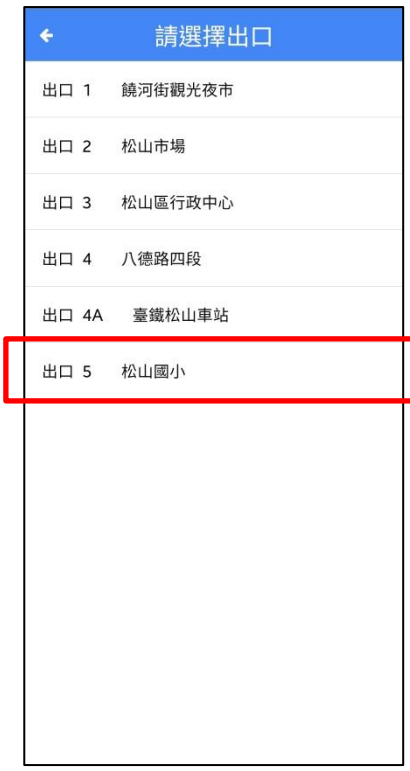


圖 67 選擇出口圖

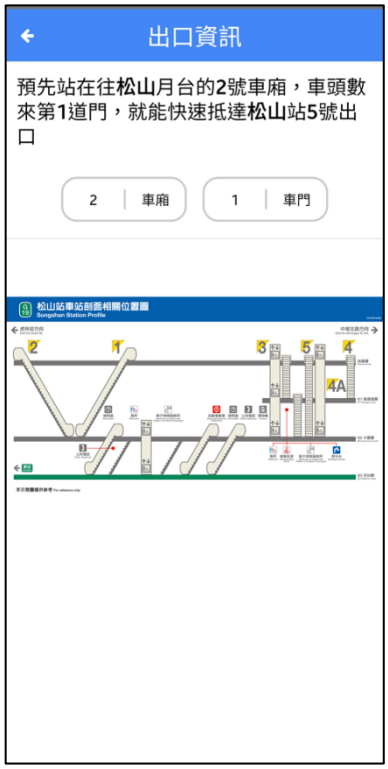


圖 68 出口資訊圖

在路線結果頁面時可點選下車資訊（圖 66 方框處），然後再點選相對的出口（圖 67）（配合圖 66 三角框處的出口號碼提示），即可得到出口資訊與圖示。

第七章 結論與未來發展

在本專題中，讓我們學習到如何用現在所學到的科技去解決所遇到的問題，透過手機 APP 軟體的搜尋功能就可直接知道所需要的相關資訊，不僅解決了問題，並且快速更有效率的完成尋找出口最近的車廂問題。有別以往的方式，讓科技帶來的便利成為了首選。

現今社會科技日新月異，這種應用 APP 系統尋找捷運出口，以新的方式展現，不但能夠讓使用者輕鬆上手，也符合現代人們講求快速便利，不僅貼心、便利，更能博得使用者的青睞。

現在手機 APP 的應用是越來越廣泛，各種大小事情都會使用手機 APP 的方式也愈來越普遍頻繁，不僅加快了整個國際社會的流動，同時對交通的查詢也更加方便。

期望未來捷出高手-北捷出口指引 APP 可以為各種使用者帶來更多的便利，不需多花體力尋找出口、更不用浪費寶貴的時間一一搜尋，既能幫使用者節省時間、也能夠解決月台人潮阻塞的問題，讓使用者只需要透過捷出高手-北捷出口指引 APP 就可以輕鬆找到指定的出口。

參考文獻

[1] 台北捷運公司，台北捷運官網，https://www.metro.taipei/

[2] 世新 Gmail 小組，Google Map API （1） 簡介，https://sites.google.com/a/mail.shu.edu.tw/wow/shi-xin-google-zhi-shi-ku/google-api/cao-zuo-shuo-ming-google-map-api-1-jian-jie

[3] 維基百科，板南線，https://zh.wikipedia.org/wiki/%E6%9D%BF%E5%8D%97%E7%B7%9A。

[4] 維基百科，松山新店線，https://zh.wikipedia.org/wiki/%E6%9D%BE%E5%B1%B1%E6%96%B0%E5%BA%97%E7%B7%9A。

[5] 維基百科，Visual Studio Code，https://zh.wikipedia.org/zh-tw/Visual_Studio_Code。

[6] Joyce Echessa，React Native 簡介：以 JavaScript 建構 iOS App（陳佳新，譯），https://www.appcoda.com.tw/react-native-introduction/。