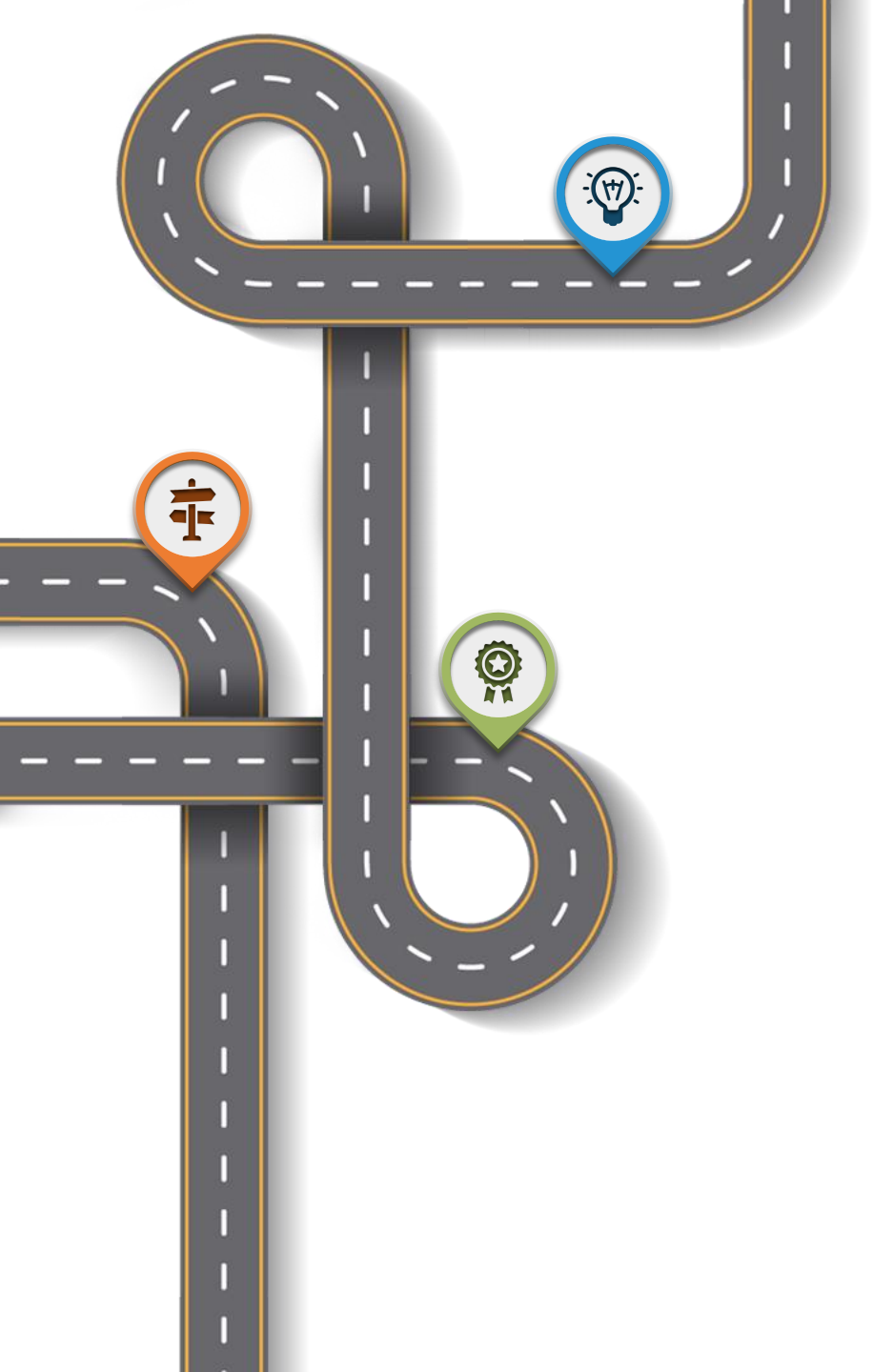
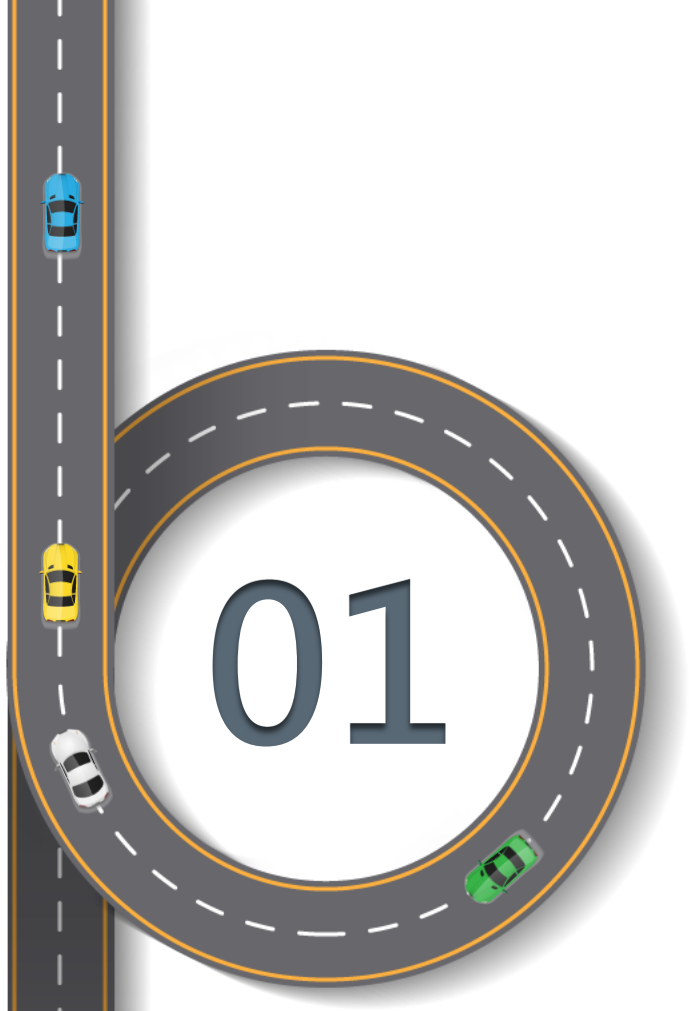
第三組

揪車Go
Just Go

組員：劉語萱、陳柏尹、廖曉珺、洪鈺姍、柳映萱



目錄



01

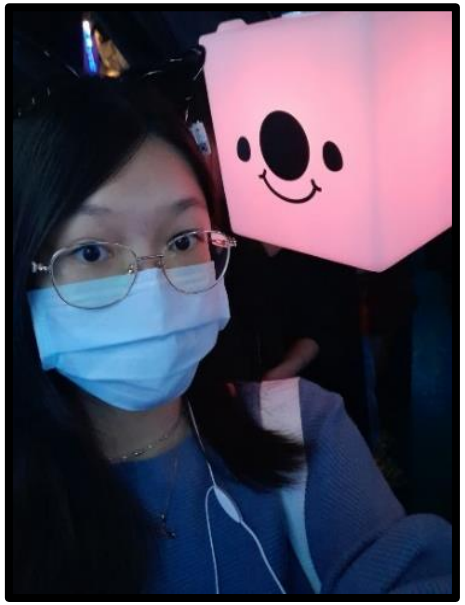
團隊介紹

01 團隊介紹

巨資一A
劉語萱
Art Designer
Associate Engineer
Line Bot Designer

巨資一A
陳柏尹

Associate Engineer
Web Designer



巨資一A
廖曉珺

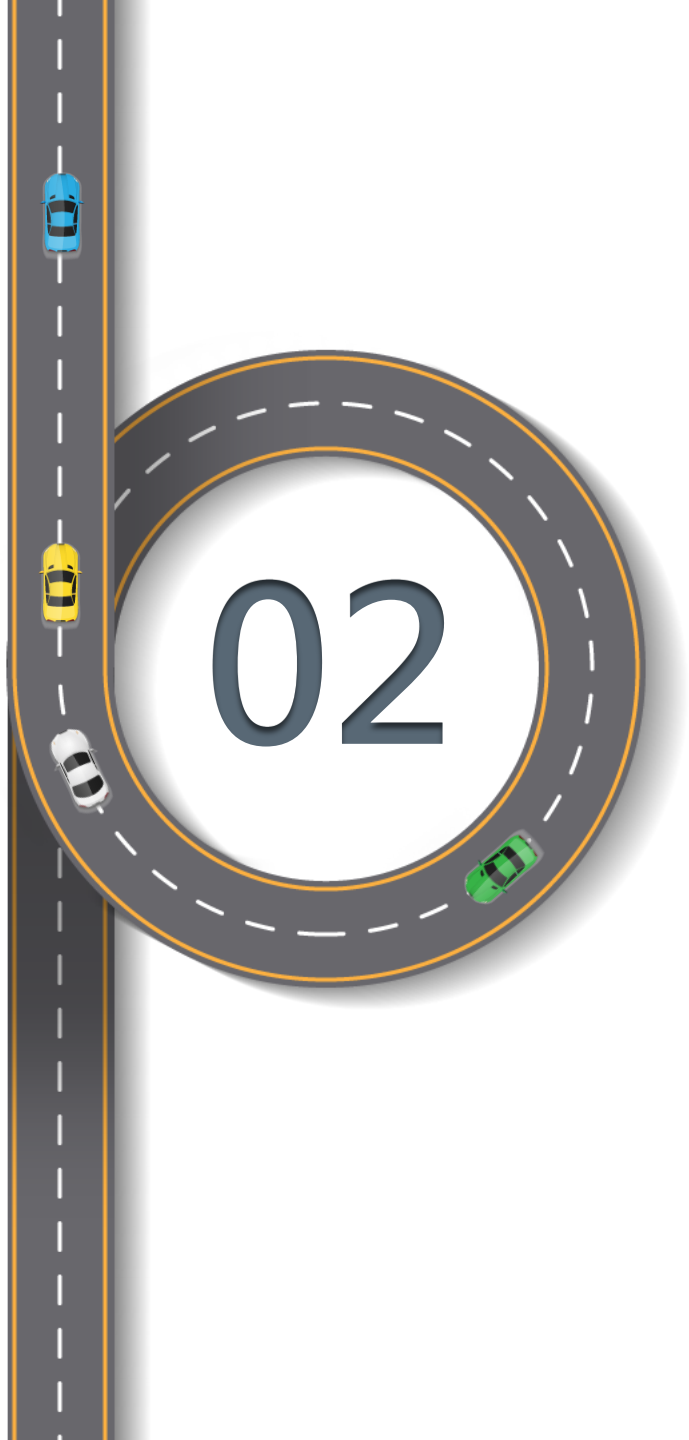
Main System Engineer

01 團隊介紹

企延B
洪鈺姍
Planner
System Engineer
Database Engineer

巨資三A
柳映萱

System Engineer



02

創作理念

01 痛點與解決方式

- **痛點：**
  1. 雖然有多班公車可到東吳外雙溪校區，但下車地點至校園仍常有段不算近的路程
  2. 部分教學大樓位於山腰，557或專車最多也只開到校內平面，要到除了校門口那幾棟外的教學大樓仍需耗費一段腳程。
  3. 校內的綜合大樓僅有的一部電梯，一旦接近上課時間排隊人潮便會增加，對於趕時間或行動較為不便的學生而言，簡直就是一大噩夢。
  4. 若以計程車作為接駁工具，能直達目標大樓，但對一般的學生而言，此選擇的負擔較大。
- **解決方式：**

  士林捷運站至東吳雙溪的車資大約落在臺幣100～140元左右，倘若可以多召集一些擁有相同目的地的學生們一起共乘，不但可以減低個人負擔，也能讓社會資源達到更有效的利用。

02 解決方式與目的

1. 製作一個供東吳學生使用的P2P平台，匹配具有相同目的地的學生。

   【目的】用較低的時間與金錢成本擁有較高的生活品質

2. 使用者可以利用看板公告的功能查看即時的共乘需求，加入共乘後可知道發起人與計程車的相關訊息

   【目的】確保基本的搭乘安全。

3. 行程結束後會自動增加1點信用值。

   【目的】顯示在刊登頁面上以提高他人的匹配意願。

03

網頁佈告欄

01 首頁

佈告欄功能：

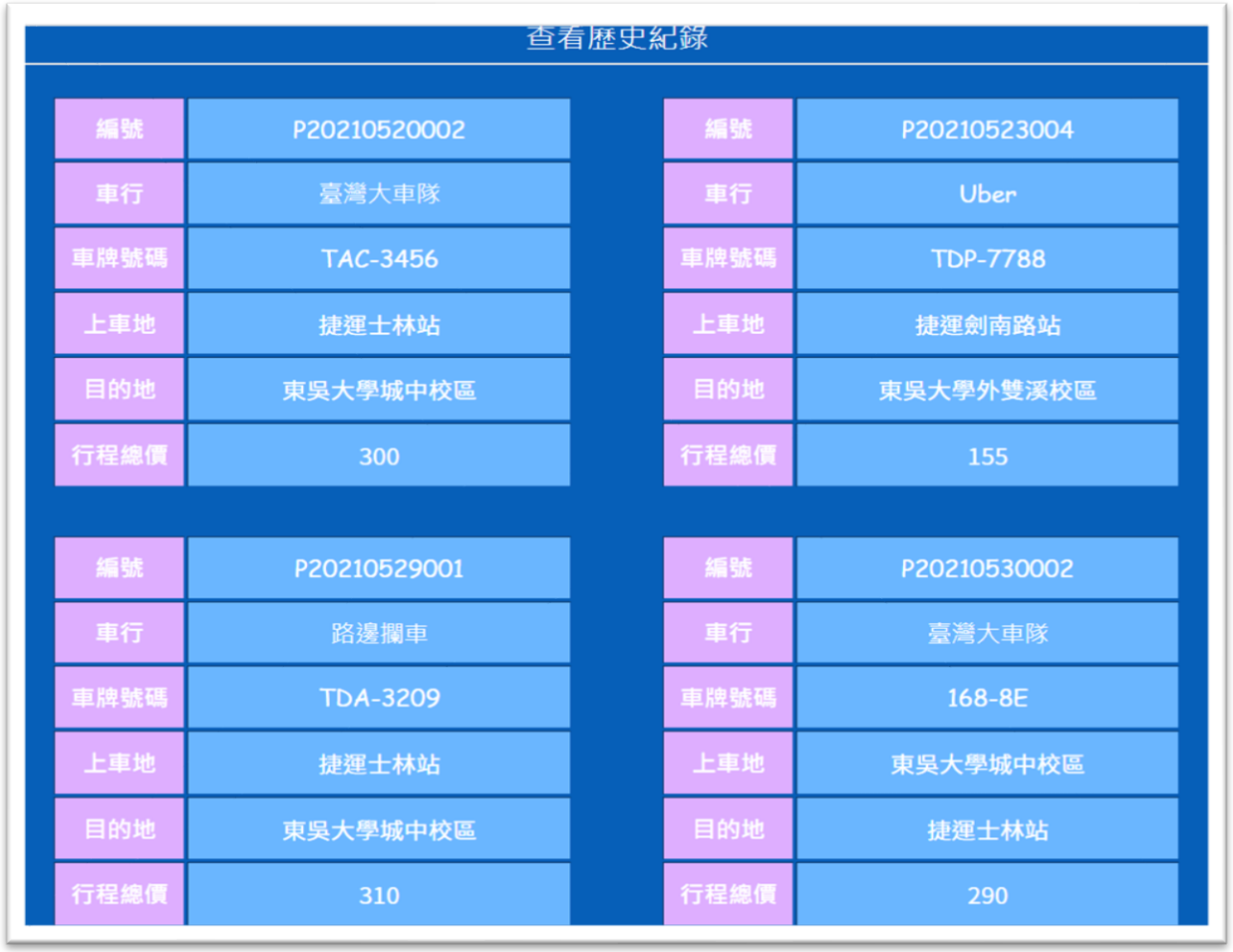
1. 顯示三項大功能的即時資訊
2. 教學影片的連結
3. 簡易Q&A問答
4. 聯絡方式

註：目前網頁是可在電腦中開啟，但因為手機上設定過多，期許未來可以將這個功能更加完善。

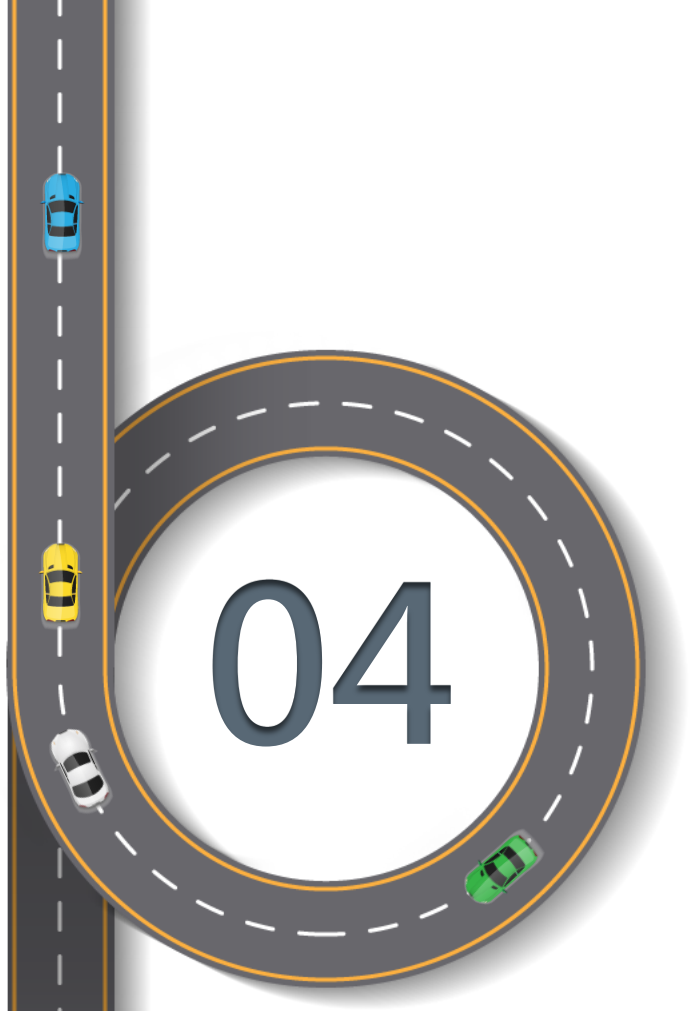
02 即時資訊

即時資訊主要是依揪車型態分流兩張表，可讓使用者依據自身需求以最快速地方式尋找到合適的刊登資料，以達到便利搭乘的目的。

03 查看歷史紀錄



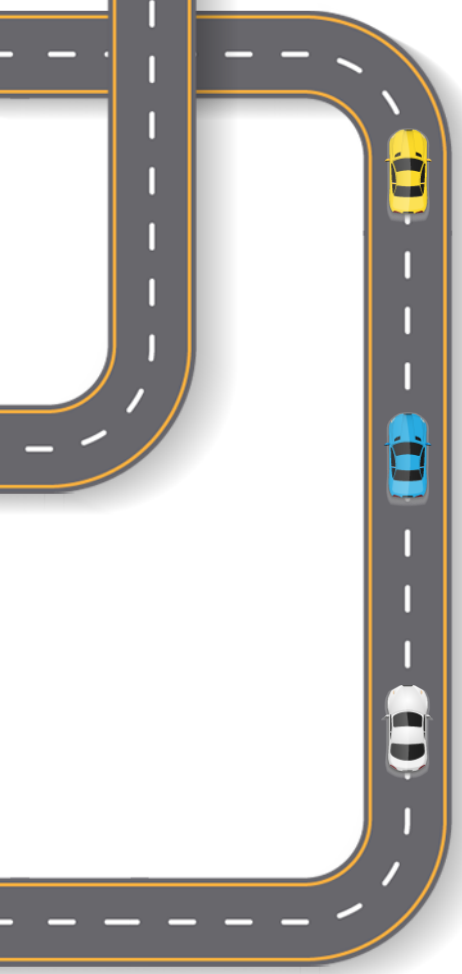
查看歷史紀錄原理是利用使用者的學號生成專屬的網址，再將此發送給使用者，因此雖然即時刊登資訊都是共享的，但僅能看到自己的歷史紀錄，以讓使用者可以查詢其搭乘的歷程記錄，資訊主要有訂單編號、車行、車牌號碼、起訖點，及行程總價。



04

程式設計流程與架構

00 介面設計流程架構



會員

註冊、更改資料、查看信用值

即時資訊

即時刊登資訊、查看歷史記錄

我要揪車

一定要揪車、湊滿人才揪車

尋找共乘

一定要揪車、湊滿人才揪車

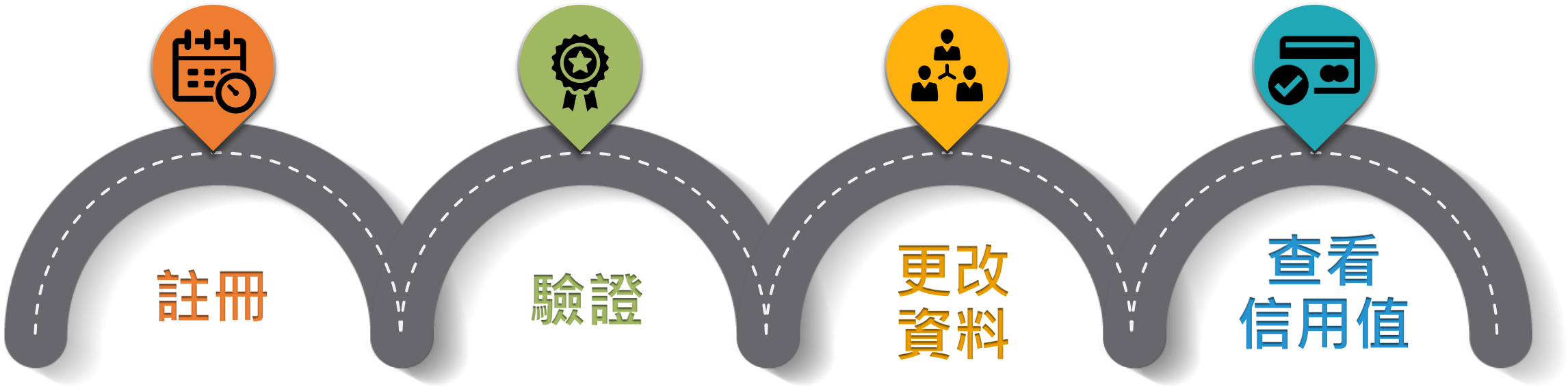
揪車情況/取消揪車

揪車現況、取消揪車、查看狀態

使用說明/QA/其他

使用說明、基本QA問答、
其他聯絡資訊、

01 會員：註冊、驗證、更改資料、查看信用值



註冊

➢ **使用流程：**

1. 會先判斷是否為新會員，若為舊會員則導引到更改資料的區塊。
2. 依序填入學號、姓名、系級、性別及密碼基本資料。

驗證

➢ **使用流程：**

1. 系統會自動送出一封email的驗證信到該位學生的東吳信箱，內含6位數的驗證碼
2. 使用者須至Line Bot上回傳驗證訊息
3. 若回傳訊息與系統相符則顯示註冊成功且資料寫入雲端資料庫，
4. 若回傳訊息與系統不相符則須重新進行註冊流程。

更改資料

更改資料大致上跟註冊相似，不同點如下：

1. 可以自行決定要從哪一項資料開始更改，順序與註冊流程相同
2. 若是初步驗證發現錯誤，則需重新進行整個更改資料的流程。
3. 更改完成後不需進行信箱驗證，直接更新資料至雲端資料庫

查看信用值

讀取資料庫內該會員的信用值，並回傳給Line Bot以發送給使用者，以利其查詢至今使用平臺後的信用值。

註：信用值越高代表乘車次數越多，亦可初步判斷為長期客戶抑或是表現優良的使用者，未來若開發福利功能，可先向此群體發送。

02 即時資訊：刊登種類、建構即時刊登的下拉式網頁html

- 依照刊登種類將訂單的即時資訊分成兩大類：
  1. 一定會發車
  2. 湊滿人才會發車

1. 可用按鈕收縮相關資訊
2. 相關資訊為一欄兩列的表格。

【註】由於是先取得當下資料庫的資訊再以Python產生一個html網頁，因此若要取得之後的即時資訊須在執行一次顯示即時資訊的流程。

03 我要揪車、尋找共乘：產生訂單的流程

當乘車者依序填完各項資訊後，我們將會特定條件向不同刊登種類的特定使用者傳送訊息。

- 一定要揪車：當達到最小乘車人數時向乘車者做最終確認。
- 湊滿人才揪車：當每多一人時向發起者告知目前人數。

當發起者依序填完各項資訊後，除了能接收到訂單的最新狀況外，還需要負責觸發訂單開始與結束的功能。

- 訂單開始：當輸入「車子已抵達」時，系統向發起人詢問實際乘車人數以判斷跑單流程。
- 訂單結束：當輸入「共乘完畢」時，系統向發起人詢問總金額並向共乘者進行確認。

04 我要揪車、尋找共乘：結束訂單的流程

當發起者輸入「共乘完畢」時，系統向發起人詢問總金額並向共乘者進行確認。

- **確認流程：**

1. 發起者輸入總金額
2. 共乘者確認每人要付的實際金額
3. 若有共乘者回復「金額_錯誤」則將回到步驟1
4. 當「金額_正確」的回覆數量與乘車者數量一樣時，系統判斷訂單完成。

當共乘者收到此訂單每人需付的金額時，需對金額進行確認。

- **確認流程：**

1. 金額正確：輸入「金額_正確」
2. 金額錯誤：輸入「金額_錯誤」

05 揪車情況/取消揪車：取消共乘/發起、查看狀態

若為取消共乘或是發起的流程，會先進入更改狀態（確認現在情況）的流程，並向使用者傳送可取消時間的訊息及警語，以達到告知取消搭乘需自行負擔後果的提醒義務，以免使用者未按取消，卻也未上車，造成信用值減分，甚至停權處置。最後即為確定是否真的放棄的提問，整個流程結束後，依然會傳送使用者狀態以讓使用者知曉目前現在自身的信用值。

用戶可以自行輸入查看狀態的服務，系統會先判定使用者帳號是否停用，並後送使用者狀態的訊息給使用者，而若非停用階段，系統會傳送訂單資訊。整體而言，查看狀態主要是在查看訂單、使用者等相關明細的流程，為了可以讓流程控制更加完善，從一開始即讓停權的使用者知道其目前無法使用服務，亦可讓搭乘的人知曉自己是否為共乘人，抑或是發起人，因為兩者的身分對於刊登的客製化些許不同，計算分擔價格，主揪者可以稍微分到些許的四捨五入小折扣。

06 使用說明/QA/其他：使用說明、基本QA問答、其他聯絡資訊

JUST
揪車GO
加額共乘 相付相乘

歡迎使用揪車just go

確定發車(已訂車)

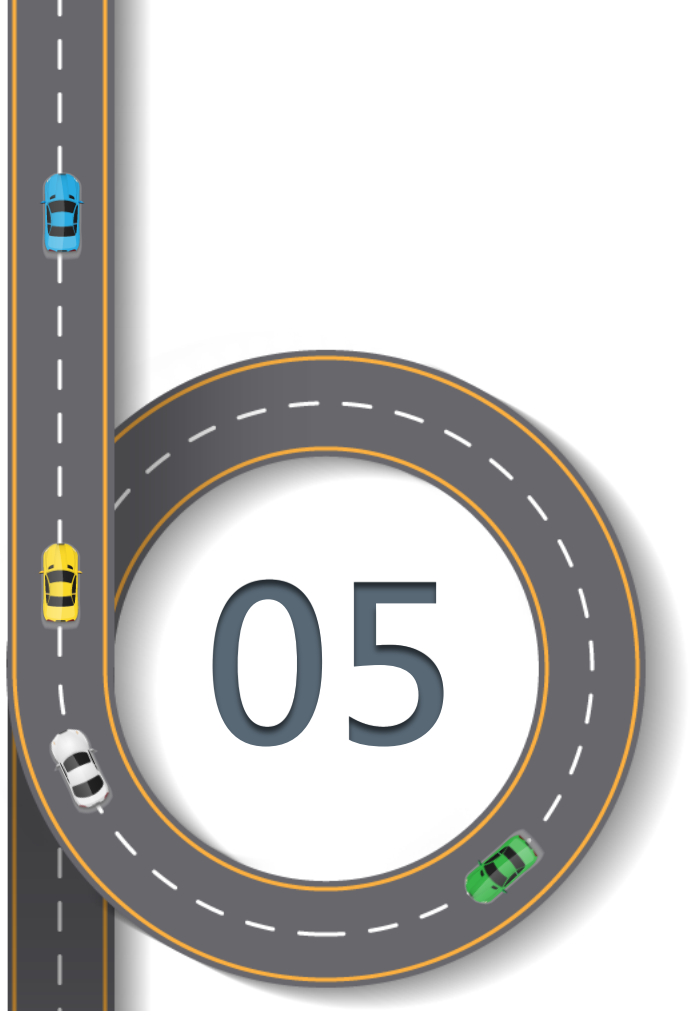
人數達標後發車(尚未訂車)

查看歷史紀錄

==使用教學==
網址：https://www.youtube.com/watch?v=hKWNcJh0EwE

==QA專區==
Q：我想取消揪車，但取消時間已過，我還能取消嗎？
A：您好，可取消時間是由發車人自行輸入的值喔！
如果超過時間，則會視為跑單，系統將扣除信用值或禁用帳號（規則如下：）
第一次跑單將會暫停用戶使用帳號一個月，第二次三個月，第三次永久封號，還請注意喔！
Q：我的學生證遺失，但我仍想認證帳號，請問有別的認證方法嗎？
A：您好，考慮到在校身分確認，我們目前暫不開放東吳大學的學校email(學號@scu.edu.tw)以外的途徑認證喔，還請見諒。

==聯絡我們==
郵件：justgocarpool@gmail.com



05

應用到的課外技術

01 利用Python傳送email給用戶

```python
import smtplib
from email.mime.text import MIMEText
from email.header import Header
from email.mime.multipart import MIMEMultipart
from email.mime.application import MIMEApplication
from string import Template
from pathlib import Path
import random
```

- 運用smtplib套件，設定SMTP的連線伺服器。
- 應用Gmail的應用程式授權金鑰
- 撰寫基本的Html以設定郵件架構，並插入附件圖片以及內含一組六位數隨機驗證碼。
- 成功發送郵件給正進行註冊的用戶進行驗證

02 利用LineBot接收與發送訊息： 將檔案連接上LineBot

```python
app = Flask(__name__)
static_tmp_path = os.path.join(os.path.dirname(__file__), 'static', 'tmp')
# Channel Access Token
line_bot_api = LineBotApi('[redacted]')
# Channel Secretgjd6kFSG3UW1X
handler = WebhookHandler('[redacted]')
global mydict
mydict={}
# 監聽所有來自 /callback 的 Post Request
@app.route("/callback", methods=['POST'])
def callback():
    # get X-Line-Signature header value
    signature = request.headers['X-Line-Signature']
    # get request body as text
    body = request.get_data(as_text=True)
    app.logger.info("Request body: " + body)
    # handle webhook body
    try:
        handler.handle(body, signature)
    except InvalidSignatureError:
        abort(400)
    return 'OK'
```

- 創立一個LINE的Messaging API專案，並取得該專案的金鑰(channel token)與密碼(channel secret)
- 將Line官方給的對接程式碼放入py檔裡，並填入金鑰與密碼
- 創立heroku帳號並將檔案git push到該專案的heroku app上
- 將該heroku app含有網域的網址(domain URL) 填入LINE的Webhook上並進行最終驗證

02 利用LineBot接收與發送訊息： LineBot處理訊息

```python
@handler.add(MessageEvent, message=TextMessage)
def handle_message(event):
    msg = event.message.text
    try:
        line_id=event.source.user_id
        line_bot_api.push_message('Uad5f20670570a6aad3f1f20c893fe5a2',TextSendMessage(text=line_id+'發了:'+msg))

    except:
        message = TextSendMessage(text="錯誤")
        line_bot_api.reply_message(event.reply_token, message)
        #line_bot_api.push_message(uu,TextSendMessage(text=uu+msg))
    if '會員' in msg:
        message = membersystem()
        line_bot_api.reply_message(event.reply_token, message)
    elif '尋找共乘' in msg:
        message = findjustgo()
        line_bot_api.reply_message(event.reply_token, message)
    elif '即時資訊' in msg:
```

- 若收到的回覆為文字訊息則將此訊息回傳給LINE帳號擁有者

- 將收到的回覆進行分流處理

02 利用LineBot接收與發送訊息： LineBot發送訊息

```
elif '註冊' in msg:
    try:
        if line_id not in mydict:
            mydict[line_id]={}
            mydict[line_id]["register"]={}
            line_bot_api.push_message(line_id,TextSendMessage(text='檢測到此id未註冊過，進行註冊程序'))
        else:
            line_bot_api.push_message(line_id,TextSendMessage(text='檢測到此id註冊過，進行更改資料程序'))


    except:
        line_bot_api.push_message(line_id,TextSendMessage(text="失敗"))
    message = register(msg, line id,mydict)
    line_bot_api.reply_message(event.reply_token,TextSendMessage(text=message))
```

- 將特定訊息回傳給特定使用者

03 架構網站：利用python生成網頁內容

```python
def web():
    if not database_connect():
        print("連線未成功 網頁未成功開啟")
    f=open('justgo.html','w',encoding='UTF-8')
    head(f)
    publish1=get_publish_D()
    sure=[]
    not_sure=[]
    for i in list(publish1):
        max_min=get_max_min(i[0])
        if max_min[0]==0:
            sure.append(i[0])
        else:
            not_sure.append(i[0])
```

用複寫檔案的方式，重新生成一個html檔

1. 抓取最新的刊登資料
2. 將資料進行分流

生成一個大表格並依資料數量生成相對應的小表格數量

```python
if len(historylist)==0:
    f.write("<span style='font-family:cwTeXYen';><font color='#FFFFFF'><h3><p>目前並無歷史紀錄<br></span>\n")
else:
    #title3(f)
    for i in range(len(historylist)):
        if i%2 ==0:
            f.write('''
<center><table border="0" width="800" style="border:0px #005AB5;">
    <tr>
        <td valign="top">\n''')
        big_form(f,left)
        history_detail=get_history(historylist[i])
        left=historyform(f,left,history_detail)
```

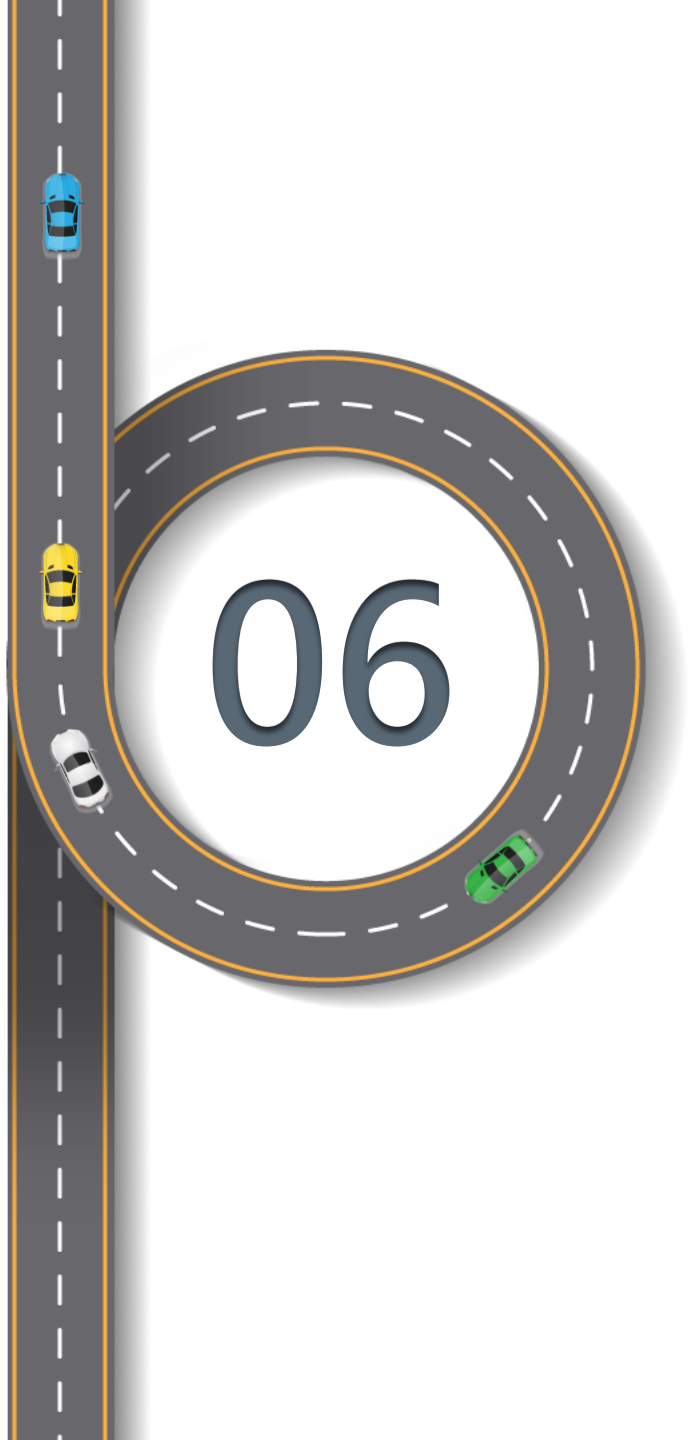
03 架構網站：利用Flask架構網站

```
conn = ps.connect(host="[redacted]",
                  user="[redacted]",
                  password="[redacted]",
                  database="[redacted]",
                  port="5432")

if conn is not None:
    cursor = conn.cursor()
    conn.commit()  # commit the changes

def get_name(uu):
    return
@app.route("/")
def home():
    return render_template("justgoexample.html",name=uu)
app.run()
```

- 將檔案與資料庫進行連線
- 利用Flask架構網站
- 將此檔案git push到該專案的heroku app上



06

Line功能使用教學 (Demo)


感謝聆聽