

Trips to the
給嚮往遠方的你
來一趟說走就走的旅行

林書妤 施樂琦 葉揚 | 程式設計第四組

瞭解更多 ►

Baseline Plan

目錄

-
- | | | |
|-------|------|------|
| 1 | 2 | 3 |
| 發想動機 | 產品受眾 | 產品介紹 |
| 4 | 5 | 6 |
| 程式碼展示 | 未來發展 | 回饋總結 |



發想動機

- 疫情後台灣人重新掀起出國熱潮。
 - 2024 年國人出國人次突破 1,684 萬，較 2023 年成長 42.8%。
- 隨機問卷結果顯示：
 - 超過 7 成旅客最在意旅遊目的地的選擇
 - 7 成 5 的人偏好自由行

痛點：沒時間規劃、不知從哪下手、資訊太多





產品受眾

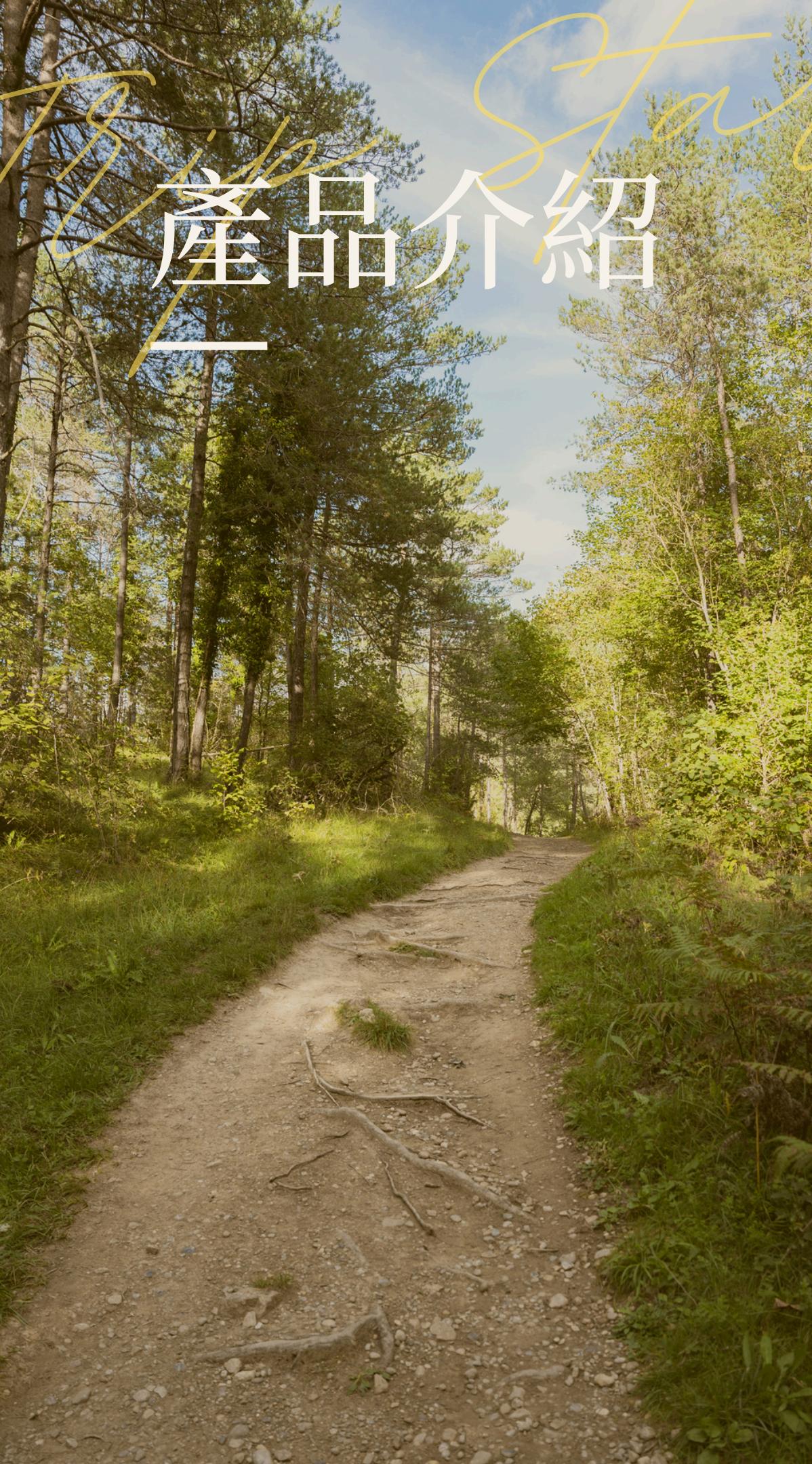
沒有規劃行程
的時間

好奇自己的
旅遊人格類型

對安排行程
毫無頭緒

想要順便搜尋其他地標

想要快速規劃
行程



產品介紹

Tip
Stay



旅遊人格測驗

找到你喜歡的旅遊模式



行程優化

使用你的旅遊風格
進行優化



自動生成行程

依照你的旅遊人格生成
適合你的行程



地點推薦

搜尋你附近的餐廳和
咖啡廳等等

程式碼展示 旅遊人格類型

```
# 旅客類型定義
TRAVELER_TYPES = [
    "探險型": {
        "description": "喜歡刺激活動、極限運動、冒險體驗",
        "keywords": ["戶外冒險", "極限運動", "登山健行", "水上活動", "刺激體驗"],
        "avoid": ["行程太緩慢", "太多購物時間"],
        "pace": "緊湊充實",
        "budget_focus": "活動門票與裝備",
        "icon": "▶"
    },
    "文化型": {
        "description": "熱愛博物館、歷史遺跡、藝術展覽",
        "keywords": ["博物館", "古蹟", "藝術展", "文化體驗", "歷史景點"],
        "avoid": ["過於商業化", "缺乏深度"],
        "pace": "適中從容",
        "budget_focus": "門票與文化導覽",
        "icon": "🏛️"
    },
    "美食型": {
        "description": "享受美食、品嚐在地佳餚",
        "keywords": ["美食", "在地文化", "品嚐", "享受"],
        "avoid": ["過於複雜", "烹飪技巧"],
        "pace": "緩慢悠閒",
        "budget_focus": "用餐與飲食",
        "icon": "🍽️"
    }
]
```

```
# 計分
type_scores = {t: 0 for t in TRAVELER_TYPES.keys()}

for q_id, answer in answers.items():
    question_data = QUESTIONNAIRE.get(q_id, {})
    options = question_data.get("options", {})

# 處理單選
    if isinstance(answer, str):
        types = options.get(answer, [])
        for t in types:
            type_scores[t] += 1
```

```
# 問卷題目設計
QUESTIONNAIRE = [
    "q1": {
        "question": "旅遊時你最期待什麼?(可複選, 最多3個)",
        "options": [
            "刺激冒險體驗": ["探險型", "自然型"],
            "深度文化探索": ["文化型"],
            "品嚐在地美食": ["美食型"],
            "血拼購物": ["購物型"],
            "放鬆身心": ["放鬆型"],
            "親近大自然": ["自然型", "探險型"],
            "體驗夜生活": ["社交型"],
            "親子同樂": ["親子型"]
        ]
    }
]
```

- 八種人格：探索、文化、美食、購物、放鬆、自然、社交、親子
- 五個問題
- 走訪答案，分數計入字典
- 排序，最高分為主要類型，次高分為次要類型

程式碼展示 行程生成

```
def get_weather_info(country, city, start_date, num_days):  
    """  
    獲取天氣資訊  
  
    Args:  
        country: 國家名稱  
        city: 城市名稱  
        start_date: 出發日期 (YYYY-MM-DD)  
        num_days: 天數  
  
    Returns:  
        dict: {  
            "success": bool,  
            "data_source": "real_api" | "fallback",  
            "weather_data": [  
    
```

```
    # ===== 4. 建構 Prompt =====  
    prompt = f"""  
    你是一位專業的旅遊規劃師, 請為以下旅客規劃 {num_days} 天的詳細旅遊行程。  
  
    **旅客資訊:**  
    - 目的地: {country} {city}  
    - 出發日期: {start_date}  
    - 旅遊天數: {num_days} 天  
    - 旅客類型: {traveler_type} {traveler_info['icon']}    - 預算範圍: {budget}  
    {f"- 特殊興趣: {interests}" if interests else ""}  
  
    **旅客類型特徵:**  
    - 描述: {traveler_info['description']}    - 偏好活動: {', '.join(traveler_info['keywords'])}  
    - 行程節奏: {traveler_info['pace']}    - 預算重點: {traveler_info['budget_focus']}    - 避免事項: {', '.join(traveler_info['avoid'])}
```

```
def get_exchange_rate(country_name):  
    """  
    取得匯率 (從 TWD 到該國貨幣)  
  
    Args:  
        country_name: 國家名稱 (例如: "日本")  
  
    Returns:  
        dict: {  
            "success": bool, # API 是否成功  
            "country": "日本",  
            "currency": "JPY",  
            "rate": 4.5, # 1 TWD = 4.5 JPY  
            "source": "api" | "default"  
        }  
    
```

- 接收使用者的目標國家、預計時間、預算
- 獲得天氣資訊
- 獲得匯率資訊
- 整合旅遊人格、旅遊資訊、天氣、匯率，建構指令給gemini
- 可獲得PDF輸出

程式碼展示 行程優化

選擇自己的步調和旅游性質

```
'planner': {'title': '⚡ 效率 (充實)', 'prompt_hint': '每天5-6個景點，緊湊', 'desc': '分秒必爭，打卡最大化'},
'balanced': {'title': '⚖ 平衡 (質感)', 'prompt_hint': '每天3-4個景點', 'desc': '景點與休息並重，享受漫遊樂趣'},
'wanderer': {'title': '📍 慢活 (度假)', 'prompt_hint': '每天2-3個景點', 'desc': '睡到自然醒，以放鬆體驗為主'},
}
pacing_choices = [info['title'] for info in pacing_data.values()]

style_data = {
    'luxury': {'title': '🏨 奢華', 'desc': '五星級住宿與米其林饗宴'},
    'nature': {'title': '🌿 秘境', 'desc': '深入山林海濱，遠離塵囂'},
    'history': {'title': '🏛 古蹟', 'desc': '漫步老街，品味歷史文化'},
    'shopping': {'title': '🛍 血拼', 'desc': '鎖定商圈，盡情享受購物'},
    'food': {'title': '🍽 美食', 'desc': '用味蕾地圖探索這座城市'}
```

可上傳初步旅游行程

```
if not file:
    return "❌ 請上傳行程檔案", None, None, None

try:
    # 讀取檔案內容
    if file.name.endswith('.pdf'):
        if WEASYPRINT_AVAILABLE:
            reader = PdfReader(file.name)
            original_text = "\n".join([page.extract_text() for page in reader.pages])
        else:
            return "❌ PDF 讀取功能不可用", None, None, None
    else:
        return "❌ 檔案格式錯誤", None, None, None
```

程式碼展示 行程優化

天氣獲取

```
# 提取城市名稱 (用於天氣查詢)
city = "Taipei"
weather_data = None
weather_summary = "無天氣資訊"

try:
    city_res = call_ai("Extract only the main city name (English) from this text: " + original_text[:500])
    clean_city = re.sub(r'^[a-zA-Z\s]', '', city_res).strip()
    if clean_city and len(clean_city) < 30:
        city = clean_city

    # 查詢天氣
    r = requests.get(
        f"https://geocoding-api.open-meteo.com/v1/search?name={city}&count=1&language=zh&format=json",
        timeout=5
    )
    if r.ok and r.json().get("results"):
        res0 = r.json()["results"][0]
        lat, lon = res0["latitude"], res0["longitude"]
        weather_data = requests.get(
            f"https://api.open-meteo.com/v1/forecast?latitude={lat}&longitude={lon}&daily=weathercode,temperature_2m_max&timezone=auto",
            timeout=5
        ).json()

        if 'daily' in weather_data:
            temps = weather_data['daily'].get('temperature_2m_max', [])
            avg_t = sum(temps)/len(temps) if temps else 25
            weather_summary = f"平均氣溫 {avg_t:.1f}°C"
    except Exception as e:
        print(f"天氣查詢失敗: {e}")

# 提取城市名稱 (用於天氣查詢)
city = "Taipei"
weather_data = None
weather_summary = "無天氣資訊"

try:
    city_res = call_ai("Extract only the main city name (English) from this text: " + original_text[:500])
    clean_city = re.sub(r'^[a-zA-Z\s]', '', city_res).strip()
    if clean_city and len(clean_city) < 30:
        city = clean_city

    # 查詢天氣
    r = requests.get(
        f"https://geocoding-api.open-meteo.com/v1/search?name={city}&count=1&language=zh&format=json",
        timeout=5
    )
    if r.ok and r.json().get("results"):
        res0 = r.json()["results"][0]
        lat, lon = res0["latitude"], res0["longitude"]
        weather_data = requests.get(
            f"https://api.open-meteo.com/v1/forecast?latitude={lat}&longitude={lon}&daily=weathercode,temperature_2m_max&timezone=auto",
            timeout=5
        ).json()

        if 'daily' in weather_data:
            temps = weather_data['daily'].get('temperature_2m_max', [])
            avg_t = sum(temps)/len(temps) if temps else 25
            weather_summary = f"平均氣溫 {avg_t:.1f}°C"
    except Exception as e:
        print(f"天氣查詢失敗: {e}")
```

獲得行程被優化的結果
獲得詳細的諮詢，如交通方式，預算，美食推薦等

****輸出格式要求****

- **行程總覽**** (含穿搭建議、必備物品)
- **每日詳細行程**** (使用 **Markdown** 表格)
 - 表格欄位：時間 | 地點 | 細項特色 | 交通方式 | 美食推薦 | 預算
 - 每個景點必須列出具體看點
 - 交通方式請具體說明 (如：捷運紅線、公車123號)
 - 美食推薦至少5項，包含價格範圍
 - 標註單項預算

可獲得PDF輸出

```
# 生成 PDF
pdf_path = None
if WEASYPYNT_AVAILABLE:
    pdf_path = "/tmp/優化後行程.pdf"
    pdf_result = convert_markdown_to_pdf_weasy(optimized_text, pdf_path)
    if not pdf_result:
        pdf_path = None
```

程式碼展示 地點推薦

地點解析與定位 (Geopy + Folium)

使用Geopy獲取指定地點的座標

```
def get_location_coordinates(country, city, landmark):
    """獲取地點座標(使用 geopy)"""

    if not GEOPY_AVAILABLE:
        return None, None, "❌ Geopy 套件未安裝"

    try:
        geolocator = Nominatim(user_agent="travel_planner")

        # 多種查詢策略
        queries = []

        if landmark:
            queries.append(f"{landmark}, {city}, {country}")
            queries.append(f"{landmark}, {city}")
            queries.append(f"{landmark}")
        else:
            queries.append(f"{city}, {country}")

        # 嘗試所有查詢策略
        for query in queries:
            try:
                location = geolocator.geocode(query, timeout=10, language='zh-TW')
                if location:
                    display_name = location.address[:60] + "..." if len(location.address) > 60 else location.address
                    return location.latitude, location.longitude, f"✅ 定位成功: {display_name}"
            except:
                continue

        return None, None, f"❌ 無法解析地點: {landmark or city}"

    except Exception as e:
        return None, None, f"❌ 座標獲取錯誤: {str(e)}"
```

使用Folium產生地圖和標記

```
def create_map_html(lat, lng, location_name, top_places_df=None, radius_km=2):
    """建立 Folium 地圖並回傳 HTML 字串"""

    if lat is None or lng is None:
        return "<div style='height: 400px; display: flex; align-items: center; justify-content: center; gap: 10px;'><img alt='Geopy logo' style='width: 20px; height: 20px;'/>地點未定位</div>"

    try:
        # 根據半徑調整地圖縮放層級
        zoom_level = radius_to_zoom(radius_km)

        # 建立地圖
        m = folium.Map(location=[lat, lng], zoom_start=zoom_level)

        # 加入原始地標標記 (紅色)
        folium.Marker(
            [lat, lng],
            tooltip=location_name,
            popup=f"📍 {location_name}",
            icon=folium.Icon(color='red', icon='info-sign')
        ).add_to(m)

        # 標註前三名餐廳
        if top_places_df is not None and not top_places_df.empty:
            # 定義前三名的顏色和圖標
            medal_icons = [
                {'color': 'orange', 'icon': 'star', 'prefix': 'fa', 'medal': 'medal-orange'},
                {'color': 'lightblue', 'icon': 'icon', 'prefix': 'fa', 'medal': 'medal-lightblue'}
            ]
```

程式碼展示 地點推薦

即時資料搜尋 (Serp API)

用Serp API搜尋附近餐廳、咖啡廳等等

```
def search_nearby_places(country, city, landmark, place_type, user_lat=None, user_lng=None):
    """搜尋附近地點"""

    # 檢查 serpapi 是否可用
    if not SERPAPI_AVAILABLE:
        return pd.DataFrame({
            "名稱": ["⚠️ serpapi 套件未安裝"],
            "評分": [0],
            "地址": ["請執行: !pip install google-search-results"],
            "距離": ["N/A"],
            "latitude": [None],
            "longitude": [None]
        })

    # 若沒有 SerpAPI Key, 返回範例資料
    if not SERPAPI_KEY or len(SERPAPI_KEY) < 10:
        return pd.DataFrame({
            "名稱": ["⚠️ 請設定 SerpAPI Key"],
            "評分": [0],
            "地址": ["無法搜尋附近地點"],
            "距離": ["N/A"],
            "latitude": [None],
            "longitude": [None]
        })
```

AI推薦 (Google Gemini API)

用Gemini API分析評分前三高景點並推薦

```
# 3 AI 推薦
recommendation = """
if gemini_client and not df.empty:
    try:
        valid_df = df_sorted[df_sorted['評分_數值'] > 0]

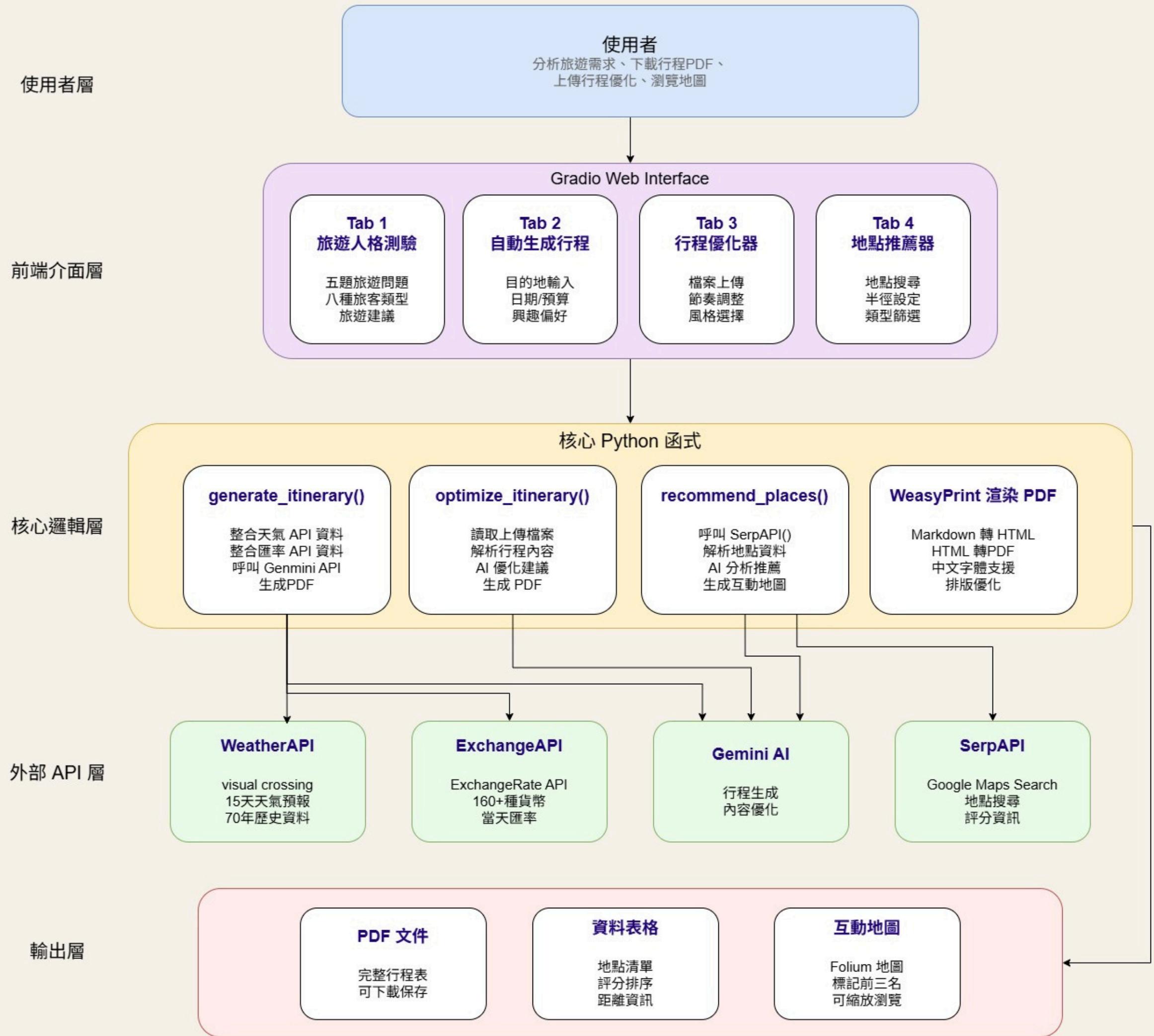
        if len(valid_df) >= 1:
            top_df = valid_df.head(3)
            # 準備給 AI 的資料(不包含座標)
            data_list = top_df[['名稱', '評分', '地址', '距離']].to_dict(orient="records")

            location_desc = f'{location_desc}, {city}' if landmark else city

            prompt = f"""
                你是美食推薦顧問。這是 {location_desc} 附近 {radius_km} 公里內評分最高的 {place_type}:
                {json.dumps(data_list, ensure_ascii=False, indent=2)}
            """

            請用 100-150 字中文, 簡潔推薦這些地點的特色。
            """
            response = model.generate_content(prompt)
            recommendation = response.text.strip()
    else:
        recommendation = "⚠ 搜尋結果中沒有評分資料, 請參考下方表格。"

```



運用之前學到的

- PDF 資料上傳與處理
 - GeminiAPI 紿建議
 - Folium 地圖
 - SerpAPI 查景點



實務效益

TripStarter 讓旅行不再是夢想清單上的待辦事項，而是隨時可以啟動的冒險。

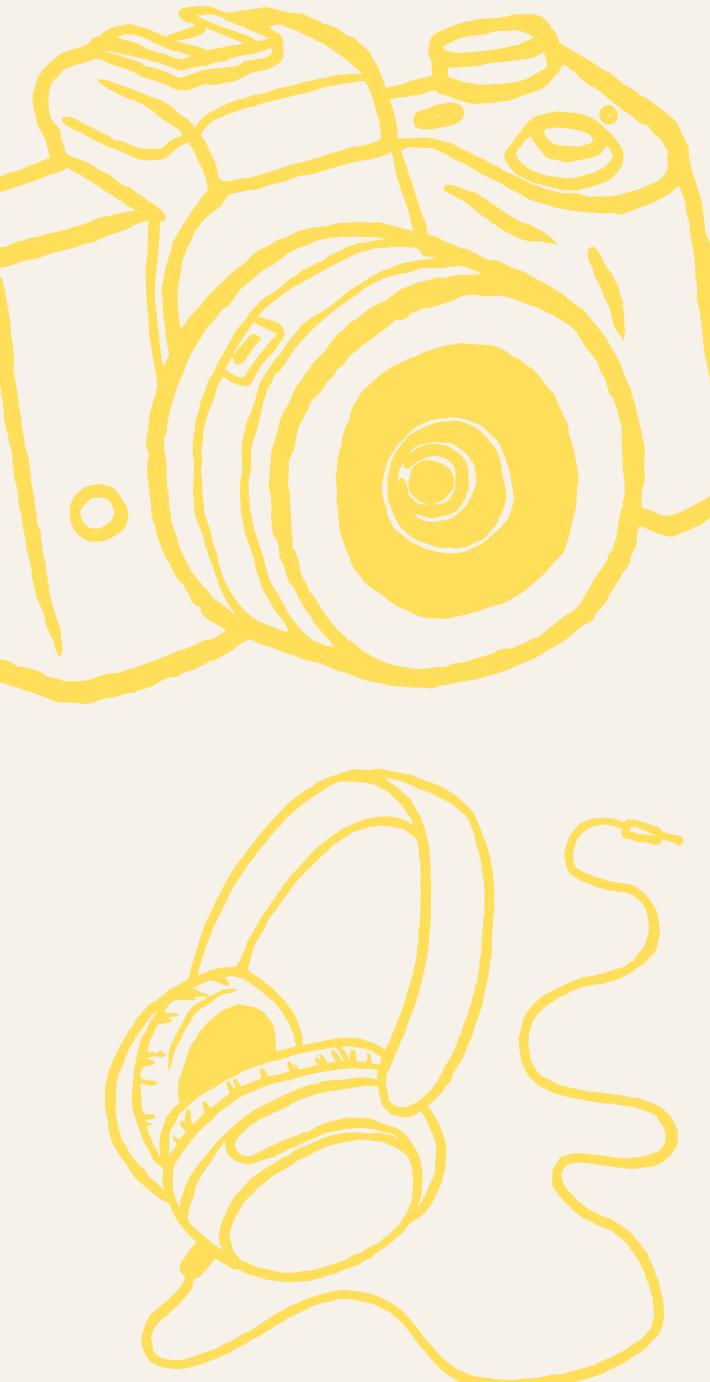
只需要一分鐘完成人格測驗，輸入你想去的國家和日期，我們就能幫你生成專屬行程。

不用再花好幾個晚上爬文、做表格，讓 TripStarter 成為你的旅行企劃師。

未來發展

- 結合飯店、航班查詢及比價功能
- 使用者登入功能
- 保留使用者旅遊歷史紀錄

回饋總結



回饋內容

完成項目或製作過程中的想法

行程規劃加入個人化需求

有類似的功能，如旅遊類型偏好

增加航班、住宿等比價

較難順利連結到相關的API
有產生API key 但資料跑不出來

以操作簡單為重

有製作方便操作的Gradio介面

增加篩選年齡層之功能

有類似的功能，如旅遊類型偏好

利用AI給「省錢密技」

讓使用者自行決定花錢模式

Trips Staffer

組內分工

人員	工作事項
林書妤	人格測驗、行程生成器、功能整合、簡報製作（程式碼展示、架構圖）
葉揚	行程優化器、簡報製作（程式碼展示）
施樂琦	地點推薦器、簡報製作（程式碼展示、其他部分）、影片剪輯

