API DOCUMENTATION & INTEGRATION GUIDE

Dự án: Smart Travel

Phiên bản: 1.0

1. DANH SÁCH API CẦN THIẾT (LIST OF APIs)

Để hệ thống Smart Travel hoạt động đầy đủ tính năng, cần tích hợp 3 nhóm API chính:

A. API Tìm kiếm Địa điểm (Location Search)

- **Mục đích:** Tìm kiếm địa điểm, lấy gợi ý khi người dùng nhập từ khóa.
- **Phương thức:** GET
- **Endpoint đề xuất:** `/api/v1/locations/search`
- **Tham số (Query Params):**
  - `q` : Từ khóa tìm kiếm (Ví dụ: "Dinh Độc Lập")
  - `limit` : Số lượng kết quả tối đa (Mặc định: 10)
- **Dữ liệu trả về (Response JSON):**

```
[
    {
        "id": "loc_123",
        "name": "Dinh Độc Lập",
        "address": "135 Nam Kỳ Khởi Nghĩa, Q1, TP.HCM",
        "type": "Di tích",
        "lat": 10.7769,
        "lng": 106.6953,
        "img": "[https://example.com/images/dinh-doc-lap.jpg](https://example.com/i
        "price": "65.000đ",
        "status": "Mở cửa"
    },
    ...
]
```

- **Gợi ý API bên thứ 3 (Hiện tại đang dùng):**
  - OpenStreetMap Nominatim: `https://nominatim.openstreetmap.org/search`

B. API Chi tiết Địa điểm (Location Details)

- **Mục đích:** Lấy thông tin chi tiết (mô tả, giờ mở cửa, giá vé...) khi người dùng chọn 1 địa điểm.
- **Phương thức:** GET

- **Endpoint đề xuất:** `/api/v1/locations/{id}`
- **Tham số (Path Param):** `id` của địa điểm.
- **Dữ liệu trả về:** Tương tự API Search nhưng đầy đủ hơn (có thêm `description`, `gallery`, `reviews` ...).

C. API Tính toán Lộ trình (Routing / Directions)

- **Mục đích:** Tính đường đi tối ưu qua nhiều điểm.
- **Phương thức:** POST
- **Endpoint đề xuất:** `/api/v1/routes/calculate`
- **Dữ liệu gửi lên (Request Body):**

```
{
    "waypoints": [
        { "lat": 10.7769, "lng": 106.6953 }, // Điểm 1
        { "lat": 10.7725, "lng": 106.6980 }, // Điểm 2
        { "lat": 10.7798, "lng": 106.6999 }  // Điểm 3
    ],
    "vehicle": "motorbike" // car, walking...
}
```

- **Dữ liệu trả về:**

```
{
    "distance": "5.2 km",
    "duration": "15 mins",
    "path": [[10.77, 106.69], [10.78, 106.70], ...] // Mảng tọa độ để vẽ đường (Poly
}
```

- **Gợi ý API bên thứ 3:** Google Directions API, Mapbox Directions, OSRM (Open Source Routing Machine).

2. HƯỚNG DẪN DÀNH CHO BACKEND DEVELOPER (INTEGRATION GUIDE)

Frontend đã được xây dựng theo mô hình "Service-Oriented", nghĩa là mọi logic gọi API đều được gom gọn trong file `js/services/api.js`.

Để kết nối Backend vào Frontend, Backend Dev chỉ cần thực hiện 3 bước đơn giản:

Bước 1: Cấu hình URL (Environment Config)

Mở file `js/config.js` và cập nhật biến `API_BASE_URL` thành địa chỉ server thực tế.

```
// Trong js/config.js
export const CONFIG = {
```

```
    API_BASE_URL: 'http://localhost:8080/api/v1', // <-- Thay đổi dòng này
    USE_MOCK_DATA: false, // <-- Chuyển thành FALSE để tắt chế độ giả lập
    // ...
};
```

Bước 2: Kích hoạt Code gọi API

Mở file `js/services/api.js`, tìm đến các hàm `getSuggestions`, `getLocationDetails`, `calculateRoute`.

- Xóa (hoặc comment lại) các đoạn logic Mock/OpenStreetMap hiện tại.
- Bỏ comment (Uncomment) các đoạn code `fetch()` đã được chuẩn bị sẵn.

Ví dụ trong hàm `getSuggestions`:

```
// Xóa đoạn này:
// const url = `https://nominatim.openstreetmap.org/...`;

// Bỏ comment đoạn này:
const url = `${this.baseUrl}/locations?q=${encodeURIComponent(keyword)}`;
const response = await fetch(url);
// ...
```

Bước 3: Mapping Dữ liệu (Quan trọng nhất)

Vì định dạng JSON của Backend có thể khác với Frontend, Backend Dev cần sửa hàm `_mapApiToApp(item)` trong `js/services/api.js` để khớp dữ liệu.

Ví dụ: Nếu Backend trả về trường `place_name` thay vì `name`:

```
// Trong hàm _mapApiToApp(item)
return {
    name: item.place_name, // <-- Sửa logic mapping tại đây
    lat: item.latitude,
    lng: item.longitude,
    // ...
};
```

-> Việc này đảm bảo Frontend không bao giờ bị lỗi hiển thị dù Backend thay đổi cấu trúc dữ liệu thế nào.

3. GHI CHÚ BỔ SUNG

- **CORS Policy:** Server Backend cần cấu hình CORS (Cross-Origin Resource Sharing) để cho phép Frontend (ví dụ: http