Project Documentation: McDonald's Outlet Finder

1. Introduction

This document provides setup instructions, key technical decisions, and essential information to understand and use the McDonald's Outlet Finder solution. The project includes functionalities for:

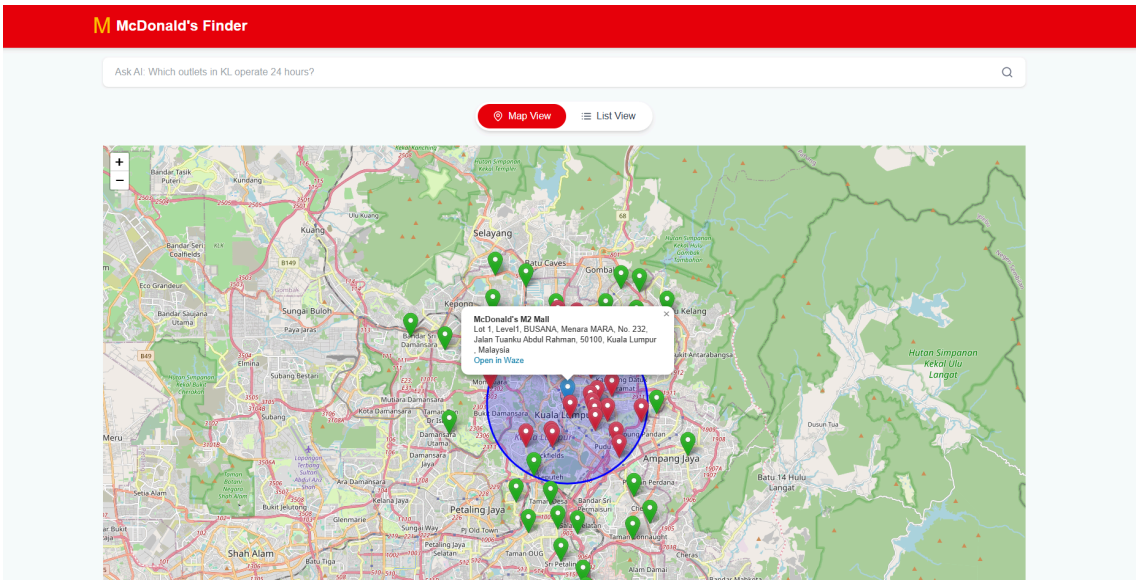
- **Web Scraping** (using McDonald's internal API
- **Geocoding** (Google Maps API for accurate coordinates)
- **API Development** (FastAPI & Supabase for data retrieval)
- **Frontend Visualization** (Next.js & Leaflet.js for interactive maps)
- **AI-Powered Search** (GPT-4o-mini for natural language queries)

2. How to Use the Website

2.1 Overview

The website provides two main views to explore McDonald's outlets in **Kuala Lumpur**:

- **Map View:** Displays all McDonald's outlets on an interactive map with a **5KM catchment radius**.
- **List View:** Shows a structured list of **all outlets**, their **addresses**, **waze link** and **available features**.



2.2 User Experience

- Upon opening the website, **all McDonald's outlets in Kuala Lumpur** are displayed by default.
- The **map view** highlights each outlet with markers.
- The **list view** presents outlet names, addresses, operating hours, and available features.

2.3 Map View Interactions

- Clicking on an **outlet marker** will:
    1. **Show a 5KM radius** around the selected outlet.
    2. **Change intersecting outlets' markers to red**, indicating they fall within the selected outlet's 5KM catchment area.
    3. **Display outlet details** (name, address, waze link) in a pop-up.
- Clicking on a different outlet will update the **5KM radius and intersecting outlets** accordingly.

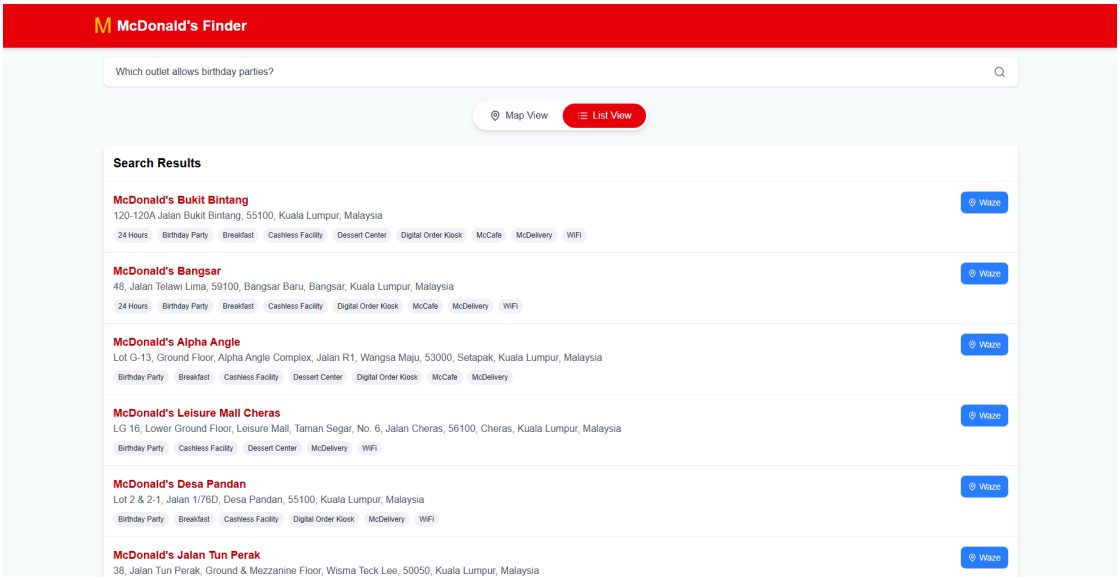
2.4 AI Search Bar

The website includes an AI-powered search bar that allows users to query outlet details based on specific features.

Example queries:

1. **"Which outlets in KL operate 24 hours?"** → Returns a list of outlets that are open 24/7.
2. **"Which outlet allows birthday parties?"** → Identifies outlets that offer birthday party services.

Users can enter natural language queries, and the AI will process the request to find relevant results.



3. Project Setup Instructions

3.1 Prerequisites

Ensure you have the following installed:

- **Python** (Backend API)
- **Node.js** (Frontend)
- **PostgreSQL** (Database - Supabase used in this project)
- **FastAPI** (Backend framework)

3.2 Backend Setup (FastAPI & PostgreSQL)

1. Clone the repository:

```
git clone "https://github.com/azibbrrrrr/webScraping.git"
cd webScraping
```

2. Start the FastAPI server:

```
uvicorn app.main:app --reload
```

3.3 Frontend Setup (Next.js & Tailwind CSS)

1. Navigate to the frontend folder:

```
git clone "https://github.com/azibbrrrrr/mcdonalds-outlets-map.git"
cd mcdonalds-outlets-map
```

2. Install dependencies:

```
npm install
```

3. Start the development server:

```
npm run dev
```

3.4 Deployment

The project is deployed to **Vercel (frontend) and Railway (backend)** for public access.

- **Frontend (Next.js):** https://outlets-map.vercel.app/
- **Backend (FastAPI):** https://web-production-c43c.up.railway.app/

4. Technical Decisions

4.1 Backend Implementation (FastAPI & Supabase)

4.1.1 Web Scraping & Database Storage (Part 1)

- Instead of traditional web scraping, the solution directly fetches JSON data from McDonald's Malaysia internal API at:

```
https://www.mcdonalds.com.my/storefinder/index.php
```

- This **removes the need for pagination handling** since the API returns **all outlets in a single request.**

The extracted data is stored in a PostgreSQL database (Supabase) with the following schema:

`mcdonalds_stores` → Stores outlet details.

`features` → Stores outlet features (e.g., 24 hours, WiFi).

`outlet_features` → Maps features to specific outlets.

```
CREATE TABLE IF NOT EXISTS mcdonalds_stores (
    id SERIAL PRIMARY KEY,
```

```
    name TEXT UNIQUE NOT NULL,
    address TEXT UNIQUE NOT NULL,
    city TEXT NOT NULL,
    state TEXT NOT NULL,
    country TEXT NOT NULL DEFAULT 'Malaysia',
    latitude DOUBLE PRECISION,
    longitude DOUBLE PRECISION,
    operating_hours TEXT,
    waze_link TEXT
);

CREATE TABLE IF NOT EXISTS features (
    id SERIAL PRIMARY KEY,
    name TEXT UNIQUE NOT NULL
);

CREATE TABLE IF NOT EXISTS outlet_features (
    outlet_id INT REFERENCES mcdonalds_stores(id) ON DELETE CASCADE,
    feature_id INT REFERENCES features(id) ON DELETE CASCADE,
    PRIMARY KEY (outlet_id, feature_id)
);
```

4.1.2 Geocoding (Part 2)

- The latitude and longitude values provided in the McDonald's API were found to be inaccurate.
- To improve accuracy, geocoding is performed using the outlet's name and address to fetch precise coordinates.
- A geocoding service,Google Maps API is used for this process.

4.1.3 API Development (Part 3)

The **FastAPI backend** serves McDonald's outlet data, enabling:

- **Fetching all outlets** (with filters for city & name).
- **Fetching a single outlet** by ID.
- **AI-powered search** using **GPT-4** to extract features from user queries.

Key API Endpoints:

- `/outlets` → Fetches all outlets (optional filters: `city`, `name`).
- `/outlets/{outlet_id}` → Retrieves **a single outlet's details**.
- `/search` → Uses AI to process **natural language queries** and return outlets based on features.

4.2 Frontend Development and Visualization (Part 4)

The frontend is built using **Next.js 13+** with **Tailwind CSS**, providing an interactive UI to visualize McDonald's outlets.

- **Map Integration**: Displays all outlets using **Leaflet.js**.
- **5KM Catchment Radius**: Clicking an outlet shows a **5KM circle** around it.

- **Intersecting Outlets Highlighted**: Outlets within overlapping 5KM radii **turn red** for better visibility.

4.3 AI-Powered Search (Part 5: Chatbot Functionality)

The search feature uses **GPT-4o-mini** to process **natural language queries** and find outlets based on requested features.

AI Query Handling Process:

1. **User enters a query** (e.g., *"Which outlets in KL operate 24 hours?"*).
2. The AI **extracts key features** from the text (e.g., `"24 Hours"`).
3. The AI **maps extracted features** to predefined **feature IDs** in the database.
4. A SQL query filters outlets that **match all requested features**.

Feature-Based Filtering:

- Each outlet has **associated features** stored in PostgreSQL.
- Example features:
    - **24 Hours** (Open all day)
    - **WiFi** (Free internet access)
    - **McCafe** (Cafe section available)
    - **Drive-Thru** (Car ordering available)

Backend Query Execution:

- If multiple features are requested (e.g., *"WiFi and McCafe"*), only outlets with **both features** are returned.
- The results are displayed on:
  Map View: Relevant outlets are highlighted.
  List View: Outlets appear in a structured list.

5. Conclusion

The **McDonald's Outlet Finder** combines **AI, mapping, and database filtering** to create an intuitive experience for users. By leveraging **FastAPI, Next.js, Leaflet.js, and GPT-4**, the project delivers an efficient solution for searching and visualizing McDonald’s outlets in Kuala Lumpur.