```
#include "GeoHandler.h"
#include "config.h"
#include <curl/curl.h>
#include "json.hpp"
#include <iostream>
#include <sstream>



// Using specific GitHub repo for json parsing
using json = nlohmann::json;

// Constructor
GeoHandler::GeoHandler(const std::string& key) : apiKey(key) {}

static size_t WriteCallback(void* contents, size_t size, size_t nmemb,
std::string* output) {
    output->append((char*)contents, size * nmemb);
    return size * nmemb;
}

// Function: Getting coordinates from zip
std::pair<double, double> GeoHandler::getCoordinates(const std::string&
zip) {
    std::string response;
    std::ostringstream url;
    url << "https://api.geoapify.com/v1/geocode/search?text=" << zip <<
"&apiKey=" << apiKey;

    // initializing Curl for http request
    CURL* curl = curl_easy_init();
    // exception handling
    if (!curl) throw std::runtime_error("Failed to init CURL");

    curl_easy_setopt(curl, CURLOPT_URL, url.str().c_str());
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteCallback);
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, &response);

    CURLcode res = curl_easy_perform(curl);
    curl_easy_cleanup(curl);
    // Exception handling if there is no response
    if (res != CURLE_OK) throw std::runtime_error("Geocoding API request
failed");

    // Parsing JSON response for location
    auto jsonData = json::parse(response);
    auto coords = jsonData["features"][0]["geometry"]["coordinates"];
    return { coords[1], coords[0] }; // lat, lon
}

// Function: Getting restaurants by location and keyword
std::vector<Restaurant*> GeoHandler::getRestaurants(double lat, double
lon, const std::string& keyword) {
    std::vector<Restaurant*> restaurants;
```

```
    std::string response;

    std::ostringstream url;
    url << "https://api.geoapify.com/v2/places?"
        << "categories=catering.restaurant"
        << "&filter=circle:" << lon << "," << lat << ",5000"
        << "&limit=5"
        << "&apiKey=" << apiKey;

    CURL* curl = curl_easy_init();
    if (!curl) throw std::runtime_error("Failed to init CURL");

    curl_easy_setopt(curl, CURLOPT_URL, url.str().c_str());
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteCallback);
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, &response);

    CURLcode res = curl_easy_perform(curl);
    curl_easy_cleanup(curl);
    if (res != CURLE_OK) throw std::runtime_error("Places API request
failed");

    auto jsonData = json::parse(response);
    for (auto& feature : jsonData["features"]) {
        auto props = feature["properties"];
        std::string name = props.value("name", "N/A");
        std::string addr = props.value("formatted", "N/A");
        std::string phone = props.value("phone", "N/A");
        std::string website = props.value("website", "N/A");

        restaurants.push_back(new Restaurant(name, addr, phone,
website));
    }

    return restaurants;
}
```