PHP:

```
<?php
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: X-API-Key, Content-Type");

function fetchChargingStations($latitude, $longitude, $apiKey) {
  $url =
"https://api.openchargemap.io/v3/poi/?latitude={$latitude}&longitude={$longitude}&di
stance=5&maxresults=5&compact=true&verbose=false&key={$apiKey}";

  $ch = curl_init();
  curl_setopt($ch, CURLOPT_URL, $url);
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
  $response = curl_exec($ch);
  curl_close($ch);

  return json_decode($response, true);
}

// Example of handling a request to fetch charging stations
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
  $latitude = $_GET['latitude'] ?? null;
  $longitude = $_GET['longitude'] ?? null;

  if ($latitude && $longitude) {
    $OPEN_CHARGE_MAP_API_KEY = 'YOUR_OPEN_CHARGE_MAP_API_KEY';
    $stations = fetchChargingStations($latitude, $longitude,
$OPEN_CHARGE_MAP_API_KEY);
```

```
    echo json_encode($stations);
  } else {
    echo json_encode(['error' => 'Latitude and longitude required']);
  }
}
?>
```

or

NODE:

```
// server.js
import express from 'express';
import fetch from 'node-fetch';

const app = express();
const PORT = 3000;

// Replace with your Mapbox and Open Charge Map API keys
const MAPBOX_ACCESS_TOKEN =
'pk.eyJ1IjoibW9oaXRoMTIiLCJhIjoiY20yd2VnNHF1MDV6azJsc2cwYXg5NWo3MyJ9.CYjk0
q98LhNWi5B7BY2vTA';
const OPEN_CHARGE_MAP_API_KEY = 'b3ff051b-042c-4646-8158-685d91c60f6e';

// Allow CORS (for development purposes only)
app.use((req, res, next) => {
  res.header("Access-Control-Allow-Origin", "*");
  res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-
Type, Accept");
  next();
});

// Route to fetch EV charging stations along a route
```

```
app.get('/api/charging-stations', async (req, res) => {
  const { source, destination } = req.query;

  try {
    // Step 1: Get route from source to destination
    const routeResponse = await fetch(`https://api.mapbox.com/directions/v5/mapbox/driving/${source};${destination}?geometries=geojson&access_token=${MAPBOX_ACCESS_TOKEN}`);
    const routeData = await routeResponse.json();

    // Check if the routeData has valid routes
    if (!routeData.routes || routeData.routes.length === 0) {
      return res.status(404).json({ error: "Route not found" });
    }

    const routeCoordinates = routeData.routes[0].geometry.coordinates;

    // Step 2: Find EV charging stations along the route
    const stations = [];

    for (const coord of routeCoordinates) {
      const [longitude, latitude] = coord;
      const stationsResponse = await fetch(`https://api.openchargemap.io/v3/poi/?latitude=${latitude}&longitude=${longitude}&distance=5&maxresults=5&compact=true&verbose=false`, {
        headers: {
          'X-API-Key': OPEN_CHARGE_MAP_API_KEY // Include the API key in the header
        }
      });
      const stationData = await stationsResponse.json();
      stations.push(...stationData); // Collect stations along the route
```

```
    }

    // Step 3: Return the collected charging stations
    res.json(stations);
  } catch (error) {
    console.error("Error fetching charging stations:", error);
    res.status(500).json({ error: "Error fetching charging stations" });
  }
});

app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Front end for executing the function:

Map.js

```
mapboxgl.accessToken =
'pk.eyJ1IjoibW9oaXRoMTIiLCJhIjoiY20yd2VnNHF1MDV6azJsc2cwYXg5NWo3MyJ9.CYjk0
q98LhNWi5B7BY2vTA'; // Replace with your Mapbox access token

// Initialize the map
const map = new mapboxgl.Map({
  container: 'map',
  style: 'mapbox://styles/mapbox/streets-v11',
  center: [78.9629, 20.5937], // Center map on India
  zoom: 5
});

// Handle the submit button click
```

```
document.getElementById('submit-btn').addEventListener('click', async function () {
  const source = document.getElementById('source').value;
  const destination = document.getElementById('destination').value;

  // Get coordinates from the source and destination
  const sourceCoords = await getCoordinates(source);
  const destinationCoords = await getCoordinates(destination);

  // Fetch and display the route
  const route = await getRoute(sourceCoords, destinationCoords);
  displayRoute(route);

  // Fetch and display EV charging stations along the route
  const stations = await getChargingStations(sourceCoords, destinationCoords);
  displayChargingStations(stations);

  // Scroll to the map section
  document.getElementById('map-section').scrollIntoView({ behavior: 'smooth' });
});

// Autocomplete functionality for the source and destination input fields
document.getElementById('source').addEventListener('input', function () {
  const query = this.value;
  if (query) {
    getSuggestions(query, 'source-suggestions');
  } else {
    clearSuggestions('source-suggestions');
  }
});
```

```
document.getElementById('destination').addEventListener('input', function () {
  const query = this.value;
  if (query) {
    getSuggestions(query, 'destination-suggestions');
  } else {
    clearSuggestions('destination-suggestions');
  }
});

// Fetch suggestions from the Mapbox Geocoding API
async function getSuggestions(query, suggestionBoxId) {
  const response = await
fetch(` https://api.mapbox.com/geocoding/v5/mapbox.places/${encodeURIComponen
t(query)}.json?access_token=${mapboxgl.accessToken}` );
  const data = await response.json();
  displaySuggestions(data.features, suggestionBoxId);
}

// Display suggestions in the dropdown
function displaySuggestions(features, suggestionBoxId) {
  const suggestionBox = document.getElementById(suggestionBoxId);
  suggestionBox.innerHTML = ''; // Clear previous suggestions
  features.forEach(feature => {
    const item = document.createElement('div');
    item.className = 'suggestion-item';
    item.innerText = feature.place_name;
    item.addEventListener('click', function () {
      document.getElementById(suggestionBoxId === 'source-suggestions' ? 'source' :
'destination').value = feature.place_name;
```

```
            clearSuggestions(suggestionBoxId);
        });
        suggestionBox.appendChild(item);
    });
    suggestionBox.style.display = features.length ? 'block' : 'none';
}

// Clear suggestions when a suggestion is selected or input is cleared
function clearSuggestions(suggestionBoxId) {
    document.getElementById(suggestionBoxId).innerHTML = '';
    document.getElementById(suggestionBoxId).style.display = 'none';
}

// Function to get coordinates from Mapbox Geocoding API
async function getCoordinates(location) {
    const response = await
fetch(`https://api.mapbox.com/geocoding/v5/mapbox.places/${encodeURIComponent(location)}.json?access_token=${mapboxgl.accessToken}`);
    const data = await response.json();
    return data.features[0].geometry.coordinates; // Return [longitude, latitude]
}

// Function to get the route from Mapbox Directions API
async function getRoute(startCoords, endCoords) {
    const response = await
fetch(`https://api.mapbox.com/directions/v5/mapbox/driving/${startCoords[0]},${startCoords[1]};${endCoords[0]},${endCoords[1]}?geometries=geojson&access_token=${mapboxgl.accessToken}`);
    const data = await response.json();
    return data.routes[0].geometry.coordinates; // Return route coordinates
}
```

```
// Function to display the route on the map
function displayRoute(route) {
  // Check if route already exists and remove it
  if (map.getSource('route')) {
    map.removeLayer('route');
    map.removeSource('route');
  }

  const routeLine = {
    type: 'Feature',
    geometry: {
      type: 'LineString',
      coordinates: route.map(coord => [coord[0], coord[1]])
    }
  };

  map.addSource('route', {
    type: 'geojson',
    data: routeLine
  });

  map.addLayer({
    id: 'route',
    type: 'line',
    source: 'route',
    layout: {
      'line-join': 'round',
      'line-cap': 'round'
```

```
    },
    paint: {
      'line-color': '#888',
      'line-width': 8
    }
  });

  map.fitBounds([[route[0][0], route[0][1]], [route[route.length - 1][0], route[route.length - 1][1]]]);
}

// Fetch charging stations from the backend API
async function getChargingStations(sourceCoords, destinationCoords) {
  try {
    const response = await fetch(`http://localhost:3000/api/charging-stations?source=${sourceCoords.join(',')}&destination=${destinationCoords.join(',')}`);

    if (!response.ok) {
      throw new Error(`Failed to fetch charging stations: ${response.statusText}`);
    }

    const data = await response.json();
    return data;
  } catch (error) {
    console.error('Error fetching charging stations:', error);
    return [];
  }
}

// Display charging stations on the map
```

```
function displayChargingStations(stations) {
  const uniqueStations = new Set();
  stations.forEach(station => {
    const coordinates = [station.AddressInfo.Longitude, station.AddressInfo.Latitude];
    const uniqueKey = coordinates.toString();

    if (!uniqueStations.has(uniqueKey)) {
      uniqueStations.add(uniqueKey);
      new mapboxgl.Marker()
        .setLngLat(coordinates)
        .setPopup(new
mapboxgl.Popup().setHTML(`<h3>${station.AddressInfo.Title}</h3><p>${station.Addre
ssInfo.AddressLine1}</p>`))
        .addTo(map);
    }
  });
}
```

MySQL(sample):

```
-- Step 1: Create the Database
CREATE DATABASE ev_charging_stations;
USE ev_charging_stations;

-- Step 2: Create the Table
CREATE TABLE charging_stations (
  id INT AUTO_INCREMENT PRIMARY KEY,
  title VARCHAR(255) NOT NULL,
  address VARCHAR(255) NOT NULL,
  latitude DECIMAL(10, 8) NOT NULL,
  longitude DECIMAL(11, 8) NOT NULL,
```

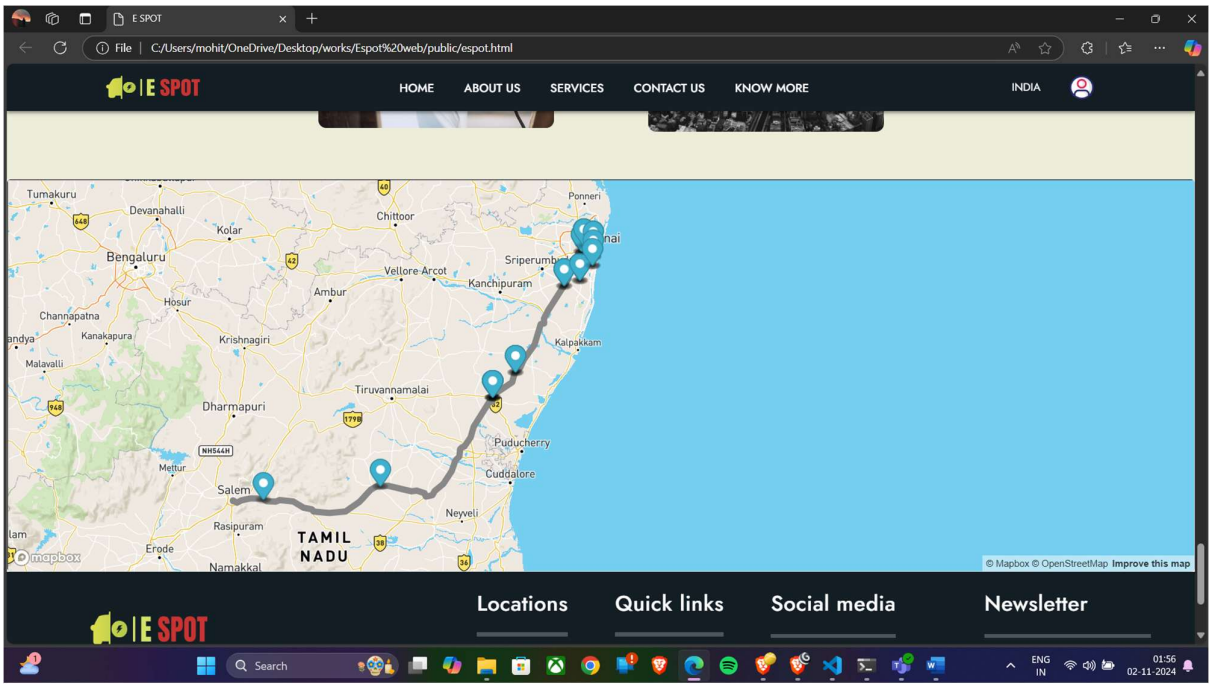
```
    company VARCHAR(100),
    charging_type VARCHAR(100)
);

-- Step 3: Insert Data
INSERT INTO charging_stations (title, address, latitude, longitude, company, charging_type) VALUES
('Charging Station 1', '123 Example St, City, State, 12345', 20.5937, 78.9629, 'Company A', 'Fast'),
('Charging Station 2', '456 Sample Rd, City, State, 67890', 21.5937, 79.9629, 'Company B', 'Standard'),
('Charging Station 3', '789 Demo Ave, City, State, 11223', 22.5937, 80.9629, 'Company C', 'Fast');

-- Step 4: Query the Data
SELECT * FROM charging_stations;
```

->note : the code is still in development.

Sample:


E SPOT
File | C:/Users/mohit/OneDrive/Desktop/works/Espot%20web/public/espot.html
E SPOT
HOME
ABOUT US
SERVICES
CONTACT US
KNOW MORE
INDIA
Tumakuru
Devanahalli
Kolar
Chittoor
Ponneri
Bengaluru
Sriperumbudur
Vellore Arcot
Kanchipuram
Ambur
Hosur
Channapatna
Kanakapura
Krishnagiri
Kalpakkam
Malavalli
Tiruvannamalai
Dharmapuri
Puducherry
Cuddalore
Mettur
Salem
Neyveli
Rasipuram
TAMIL NADU
Erode
Namakkal
mapbox
© Mapbox © OpenStreetMap Improve this map
E SPOT
Locations
Quick links
Social media
Newsletter
Search
ENG IN
01:56 02-11-2024