TECHNICAL DOCUMENTATION: LUMBINI PROVINCE TOURIST EXPLORER

Arbin Chaudhary, Nitesh Yadav, Bhumika Singh
arbinchaudhary.0006@gmail.com, ny5051145@gmail.com, imbhumeeka@gmail.com

Table of Contents

Overview

The **Lumbini Province Tourist Explorer** is a web-based interactive mapping application built using **Leaflet.js** to showcase tourist attractions in Lumbini Province, Nepal. It allows users to explore tourist places, filter by district, view routes from Tribhuvan International Airport (TIA) or between selected places, display buffers around points of interest, and visualize administrative boundaries (states, districts, and municipalities). The application integrates **GeoJSON** data for tourist locations and boundaries, and uses external libraries like **Leaflet Routing Machine**, **Turf.js**, and **Leaflet Control Geocoder** for enhanced functionality.

Features

- **Interactive Map**: Displays tourist attractions in Lumbini Province with markers and permanent labels.

- **District Filtering**: Filter attractions by selecting a district from a dropdown.

- **Place List**: Displays clickable list of places for the selected district.

- **Routing**: Calculate and display routes from TIA or between two selected places using Leaflet Routing Machine.

- **Buffer Analysis**: Generate a 5 km buffer around a selected attraction using Turf.js.

- **Geocoding**: Search for locations using Leaflet Control Geocoder.

- **Administrative Layers**: Toggle visibility of Nepal's state, district, and municipality boundaries.

- **Layer Control**: Switch between base maps (OpenStreetMap, ESRI Satellite, Topo Map) and overlay layers.

- **Responsive Design**: Sidebar and map layout adapts to different screen sizes.

Technologies Used

- **HTML5** : Structure of the web page.

- **CSS3** : Styling for the UI, including custom gradients and responsive design.

- **JavaScript** : Core logic for map interactions and dynamic content.

- **Leaflet.js** : Open-source library for interactive maps.

- **Leaflet Routing Machine** : For route calculation and visualization.
- **Turf.js** : For geospatial analysis (e.g., buffer creation).
- **Leaflet Control Geocoder** : For location search functionality.
- **GeoJSON** : Format for storing tourist attractions and administrative boundaries.
- **External APIs** :
  - OpenStreetMap, ESRI Satellite, and OpenTopoMap for base map tiles.
  - Nominatim (via Leaflet Control Geocoder) for geocoding.

System Architecture

The application is a single-page web application running entirely in the browser. It fetches GeoJSON data for tourist places and administrative boundaries from local files (`data/lumbini_places.geojson`, `data/nepal-districts-new.geojson`, `data/nepal-states.geojson`, `data/nepal-municipalities.geojson`). The Leaflet map is initialized with multiple base layers and overlays, and user interactions (e.g., district selection, routing, buffering) trigger dynamic updates to the map and sidebar.

Workflow

1. **Initialization** : Load map with default view centered on Lumbini Province ([27.5, 83.5], zoom level 8).
2. **Data Loading** : Fetch and parse GeoJSON files for tourist places and boundaries.
3. **User Interaction** : Handle events like district selection, place clicks, route requests, and buffer creation.
4. **Rendering** : Update map markers, layers, and sidebar content based on user actions.
5. **External Services** : Use Nominatim for geocoding and OSRM (via Leaflet Routing Machine) for routing.

File Structure

```
├── index.html                  # Main HTML file
├── data/
│   ├── lumbini_places.geojson  # Tourist attractions data
│   ├── nepal-districts-new.geojson  # District boundaries
│   ├── nepal-states.geojson    # State boundaries
│   ├── nepal-municipalities.geojson  # Municipality boundaries
```

Setup and Dependencies

Prerequisites

- A modern web browser (e.g., Chrome, Firefox, Edge).

- Internet connection to load external libraries and map tiles.

Dependencies

The application uses the following CDNs:

- Leaflet.js: `https://unpkg.com/leaflet/dist/leaflet.js`

- Leaflet CSS: `https://unpkg.com/leaflet/dist/leaflet.css`

- Leaflet Routing Machine: `https://unpkg.com/leaflet-routing-machine/dist/leaflet-routing-machine.js`

- Leaflet Routing Machine CSS: `https://unpkg.com/leaflet-routing-machine/dist/leaflet-routing-machine.css`

- Turf.js: `https://unpkg.com/@turf/turf@6.5.0/turf.min.js`

- Leaflet Control Geocoder: `https://unpkg.com/leaflet-control-geocoder/dist/Control.Geocoder.js`

- Leaflet Control Geocoder CSS: `https://unpkg.com/leaflet-control-geocoder/dist/Control.Geocoder.css`

Setup Instructions

1. Place the `index.html` file and `data/` folder in a web server directory (e.g., Apache, Nginx, or a local server like `python -m http.server`).

2. Ensure the GeoJSON files are correctly placed in the `data/` directory.

3. Open `index.html` in a browser to view the application.

Data Sources

- **lumbini_places.geojson** : Contains 20 tourist attractions in Lumbini Province, each with properties (`name`, `type`, `district`, `info`, `image` or `image_source`) and point geometry (longitude, latitude).

- **nepal-districts-new.geojson** : District boundaries for Nepal.

- **nepal-states.geojson** : State boundaries for Nepal.

- **nepal-municipalities.geojson** : Municipality boundaries for Nepal.

Each tourist place includes:

- **name** : Name of the attraction (e.g., "Maya Devi Temple").

- **type** : Category (e.g., Religious, Historical, Natural).

- **district** : District in Lumbini Province.

- **info** : Brief description.

- **image** or **image_source** : URL to an image of the attraction.

User Interface

Sidebar

- **Title** : "NP Lumbini Tourist Map".

- **District Dropdown** : Select a district to filter attractions (e.g., Rupandehi, Kapilvastu, or "All").

- **Place List** : Displays clickable names of attractions for the selected district.

- **Route Form** :

  - Dropdowns for selecting "From" and "To" places.

- "Show Route Between Places" button to display a route.

- "Clear Route" button to remove the route and restore markers.

Map

- **Base Layers** : OpenStreetMap (default), ESRI Satellite, Topo Map.
- **Overlay Layers** : District, State, and Municipality boundaries (toggleable).
- **Markers** : Represent tourist attractions with permanent tooltips showing names.
- **Popups** : Display details (name, type, district, info, image) and buttons for routing and buffering.
- **Geocoder** : Search bar for finding locations.
- **Routing Control** : Displays routes with a close button.
- **Spinner** : Loading animation during route calculation.

Functionalities

1. **Map Initialization** :
   - Centered at [27.5, 83.5] with zoom level 8.
   - Default base layer: OpenStreetMap.
2. **District Filtering** :
   - Selecting a district filters attractions and updates the place list and map markers.
   - "All" option shows all attractions.
3. **Place Interaction** :
   - Clicking a place in the list centers the map on it and opens its popup.
   - Markers have permanent tooltips with names.
4. **Routing** :
   - "Show Route from TIA" in popups calculates a route from TIA ([27.6971, 85.3596]).
   - "Show Route Between Places" calculates a route between two selected places.

- Routes are displayed with red and yellow lines, and markers are hidden during routing.
- A close button removes the route and restores markers.

5. **Buffer Analysis** :
   - "Show Buffer (5 km)" creates a 5 km buffer around a selected point using Turf.js.
   - "Clear Buffer" removes the buffer.
6. **Geocoding** :
   - Search for locations using Nominatim, adding a marker with a label for results.
7. **Layer Control** :
   - Toggle between base maps and overlay layers (districts, states, municipalities).

Code Breakdown

HTML

- **Structure** : A sidebar (` sidebar`) for controls and a map container (` map`).
- **Elements** :
  - Dropdowns for district and route selection.
  - Buttons for routing and clearing routes.
  - Place list (` placeList`) for filtered attractions.
  - Spinner (` spinner`) for loading states.

CSS

- **Layout** : Flexbox for sidebar and map.
- **Styling** :
  - Custom gradients for buttons and place list items.
  - Responsive design with shadows, borders, and hover effects.
  - Custom marker labels and routing container styles.
- **Animations** : Spinner animation for loading.

JavaScript

- **Map Setup** :
  - Initializes Leaflet map with base layers (OpenStreetMap, ESRI, Topo).
  - Adds geocoder control with a custom search button.
- **Data Loading** :
  - Fetches `lumbini_places.geojson` and populates markers and dropdowns.
  - Fetches boundary GeoJSON files and adds them as overlay layers.
- **Event Handlers** :
  - `districtSelect`: Filters places and updates the map and place list.
  - `placeList` clicks: Centers map and opens popup.
  - `showRouteBtn`: Calculates route between selected places.
  - `clearRouteBtn`: Removes route and restores markers.
  - `routeBtn` (in popup): Routes from TIA.
  - `bufferBtn` (in popup): Creates 5 km buffer.
  - `clearBufferBtn` (in popup): Removes buffer.
  - `geocoder markgeocode`: Adds search result marker.
- **Routing** :
  - Uses Leaflet Routing Machine with OSRM for route calculation.
  - Custom line styles (red outline, yellow core).
  - Adds a close button to the routing container.
- **Buffer Analysis** :
  - Uses Turf.js to create a 5 km buffer around a point.
  - Styles buffer with blue outline and light blue fill.
- **Layer Control** :
  - Adds toggleable base and overlay layers after boundaries are loaded.

GeoJSON Data

- **lumbini_places.geojson** : 20 features with point geometries and properties.

- **Boundary Files** : Polygon geometries for districts, states, and municipalities, styled with distinct colors and weights.

Styling

- **Color Scheme** :

  - Primary: Purple gradients (e0c3fc, 7a55ff) for place list, buttons, and form.

  - Secondary: Red (d62828) for headings and spinner.

  - Accent: Blue (1976d2, 90caf9) for buffers, green (009688) for municipalities, orange (ff7800) for states.

- **Typography** : Segoe UI for consistency.

- **Effects** : Shadows, transitions, and gradients for a modern look.

- **Marker Labels** : Semi-transparent white with purple borders, permanently displayed.

- **Routing Lines** : Red outline with yellow core for visibility.

Future Improvements

- **Dynamic Buffer Radius** : Add an input field for custom buffer distances.

- **Hotel Search** : Integrate OpenStreetMap Overpass API to fetch and display nearby hotels.

- **Offline Support** : Cache GeoJSON and map tiles for offline use.

- **Mobile Optimization** : Enhance touch interactions and responsive layout.

- **Error Handling** : Improve error messages for failed fetches or routing.

- **Additional Data** : Include more attractions and detailed information.

- **Accessibility** : Add ARIA labels and keyboard navigation.

Known Issues

- **GeoJSON Fetch Errors** : If GeoJSON files are missing or malformed, the map may not load correctly.
- **Routing Limitations** : OSRM may not always find routes for remote locations.
- **Buffer Performance** : Large buffers may slow down the map on low-end devices.
- **Geocoder Dependency** : Relies on Nominatim, which may have rate limits.
- **Image Loading** : Some image URLs may be slow or unavailable.