1. Introduction
Mentese Navigator is a JavaScript-based pathfinding and navigation application developed for the Mentese district of Mugla. The application allows users to select multiple stops on an interactive map and calculates the most efficient route between these points using Dijkstra’s Algorithm. The mapping interface is powered by the Leaflet.js library.

2. Core Logic & Algorithm (dijkstra.js)
This file serves as the "brain" of the project, handling the shortest path calculations based on graph theory principles.

- findShortestPath(graphData, startNode, endNode): This is the primary function of the algorithm.

  -Initialization: It sets the initial distance of all nodes to infinity, except for the starting node, which is set to 0.

  -Dijkstra Iteration: The algorithm continuously selects the node with the smallest distance from the remainingNodes set and explores its neighbors.

  -Distance Update (Relaxation): If a new path to a neighbor is found to be shorter than the previously recorded distance, the distance is updated, and the current node is saved in the previous list.

  -Path Reconstruction: Once the destination node is reached, the algorithm backtracks through the previous list to reconstruct the optimal path.

3. Map Interaction & Management (script.js)
This script manages visual interactions, map rendering, and user inputs.
- Map Initialization: The map is centered on Menteşe using the MENTESE_COORDS variable, loading OpenStreetMap (OSM) tiles.
- Node Selection: Using the map.on('click', ...) function, the app captures coordinates from user clicks. The findNearestNode() function then maps these coordinates to the closest real-world road node in the dataset.
- Multi-Stop Support: Each user click adds a new marker (Stop 1, Stop 2, etc.), which is stored in the selectedNodes array.
- calculateFullRoute(): This function connects all selected stops sequentially (e.g., from Stop 1 to Stop 2, then Stop 2 to Stop 3). It calculates each segment individually, renders the polyline on the map, and displays the total accumulated distance.

4. Data Structure (graph-data.json)
The application processes the road network as a "Graph" structure:
- Nodes: Represent intersections and road split points with unique IDs.
- Coordinates: Store the Latitude and Longitude for each node to enable map placement.
- Edges: Define the connections between nodes, including the weight (distance in meters) required for Dijkstra's calculations.

5. Frontend & UI (index.html & style.css)
- Sidebar: Provides instructions to the user, tracks the number of added stops, and displays the final calculated distance.
- Map Container: A full-screen container where the interactive map and routes are rendered.
- Responsive Design: Styled with CSS to ensure a user-friendly layout where the sidebar and map are properly aligned and accessible.