VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belgaum-590014



A PROJECT REPORT ON

**“GPS NAVIGATOR USING DIJKSTRA’S ALGORITHM”**

Submitted to

**Visvesvaraya Technological University**

In the partial fulfilment of requirements for the award of degree

**BACHELOR OF ENGINEERING** IN

**INFORMATION SCIENCE AND ENGINEERING**

BY

**Dhruva Bhat S N 4CB21IS014**

**Shivaprada 4CB21IS047**

Under the Guidance of **Prof. Shilpa B**

Dept of IS&E

**Canara Engineering College,**

**Department of Information Science and Engineering**

**Benjanapadavu – 574219**

**2022 – 23**

**CANARA ENGINEERING COLLEGE**

**Benjanapadavu-574219**

**2022-2023**

**Department of Information Science and Engineering**



**CERTIFICATE**

This is to certify that the major project work entitled **“GPS NAVIGATOR USING DIJKSTRA’S ALGORITHM”** is a bonafide work carried out by **Mr. DHRUVA BHAT S N** bearing **4CB21IS014** and **SHIVAPRADA** bearing **4CB21IS047**,the bonafide students of **Canara Engineering College** in partial fulfilment for the award of degree of Bachelor of Engineering in Information Science and Engineering under the **Visvesvaraya Technological University, Belagavi** during the academic year **2022-2023.**It is verified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The major project report has been approved as it satisfies all the academic requirements in the respect of major project work prescribed by the Bachelor of Engineering Degree.

|  |  |  |
| --- | --- | --- |
| **Signature of Project Guide**  **Mrs. Shilpa B**  Assistant Professor | **Signature of HOD**  **Dr. Jagadisha N**  Associate Professor & HOD | **Signature of Principal**  **Dr. Ganesh V Bhat**  Principal |

**Name of the Examiner Signature with date**

1)

2)

##### **ABSTRACT**

The GPS Navigator with Shortest Path Computation and Graph Visualization is a web-based application that facilitates efficient route planning and visualization on digital maps. The project aims to provide the user with a user-friendly interface to find the shortest path between two locations on a map, leveraging Dijkstra’s algorithm. Additionally, it offers a graphical representation of the computed path, along with the traffic details, to enhance the user experience.

Key Words: GPS Navigator, web-based application, route planning, shortest path, Dijkstra’s algorithm, road network, weighted graph, map visualization.

## ACKNOWLEDGEMENT

A successful and satisfactory completion of any significant task is the outcome of a valuable aggregate combination of different people in a radial direction explicitly and implicitly. We have been lucky to have received a lot of help and support from our lecturers during the making of this project, we would therefore take the opportunity to thank and express our gratitude to all those without whom the completion of our project would not be possible. We owe a great thanks to Principal **Dr. Ganesh V. Bhat**, for providing his kind support and cooperation. We are extremely grateful to **Dr. Jagadisha N**, Head of the Information Science & Engineering Department for his moral support and encouragement. We consider it a privilege and honor to express our sincere gratitude to our project guide **Prof. Shilpa B** for guiding and improving our knowledge of this work.

We consider it a privilege and honor to express our sincere gratitude to our project faculties for their cooperation to complete this mini project. Sushma M D who helped us in doing web-based application and all teaching and non-teaching staff of the Department of IS&E for their valuable support and encouragement.

|  |  |
| --- | --- |
| **Project Associates** |  |
| **Dhruva Bhat S N** | **4CB21IS014** |
| **Shivaprada** | **4CB21IS047** |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Contents | Page Number |
| Introduction | 1 |
| Problem Statement | 2 |
| Methodology | 4 |
| Design | 6 |
| Implementation | 8 |
| Result | 13 |
| Conclusion |  |

##### **INTRODUCTION**

The GPS Navigator project is a web-based application to simplify route planning and provide users with the shortest and most efficient path between two locations on a map. The system utilizes advanced algorithms, particularly Dijkstra’s algorithm, to compute the optimal route in a road network represented as a weighted graph with non-negative edge weights.

In today’s fast-paced world, efficient navigation and route optimization have become essential for travelers, commuters, and delivery services. Traditionally paper maps and static directions are often insufficient to cope with the complexity of modern road networks and real-time traffic conditions. As a result, digital navigation system has become increasingly popular due to their ability to dynamically calculate optimal routes, saving time and resources for users.

The GPS Navigator project addresses these navigation challenges by providing an intuitive and user-friendly web-application. Users can interact with the map interface, select a starting point and destination, and visualize the shortest path instantly. Behind the scenes, the application leverages Dijkstra’s algorithm to calculate the optimal path considering the distances and weights associated with the road network’s edges.

**Key Features:**

* Interactive Map Interface
* Shortest Path Calculation
* Real-time Traffic Simulation
* Visualizing the Route
* Graph Visualization

##### **PROBLEM STATEMENT**

* + - Implement a GPS Navigator to show the shortest path between the two paces using Dijkstra’s algorithm and create a graph visualization based upon the traffic conditions. Make it user-friendly and host it.
    - The GPS Navigator project aims to address the challenges and complexities associated with route planning and navigation system often lack the ability to efficiently compute optimal routes considering real=time traffic conditions and diverse transportation options. The absence of an intuitive and user-friendly platform for visualizing the shortest path between two locations limits users’ ability to make informed decisions regarding their travel plans.

### PURPOSE

The following are the purpose of the proposed project:

* + - To develop an advanced web-based GPS navigation system that assists users in efficient route planning and navigation.
    - To offer a user-friendly platform that simplifies the process of finding the shortest and most optimal path between two locations on a map.
    - To provide real-time traffic updates and estimated travel times to enhance decision-making during route selection.
    - To improve overall travel efficiency and reduce travel times for users in a variety of transportation scenarios.

### OBJECTIVE

* + - Implementing Dijkstra’s algorithm to calculate the shortest path in a weighted graph representing road network.
    - Develop an interactive ap interface that allows users to select start and end locations by clicking on the map.
    - Incorporate real-time traffic simulation to provide users with estimated travel times based on the current traffic conditions.
    - Create clear visualizations of the selected route on the map, enhancing users’ understanding and decision-making.
    - Design a user-friendly and intuitive interface that ensures a smooth navigation experience for all users.
    - Host the web-application.

### EXPECTED OUTCOME

* + - A fully functional GPS navigation web application capable of efficiently calculating the shortest path between two locations.
    - An interactive map interface enabling users to easily select their start and end points for route planning.
    - Real-time traffic updates and estimated travel times displayed to users, enabling them to make informed decisions.
    - Clear visualization of selected routes on the map ensuring users have a comprehensive view of their intended journey.
    - A user-friendly and intuitive platform that enhances overall travel efficiency and provides an improved navigation experience

##### **METHODOLOGY**

**1. Project Setup:**

* Set up the development environment with necessary tools, including HTML, CSS, JavaScript, and Leaflet.js library for map integration.
* Obtain API keys or access tokens for the mapping service and traffic data provider.

**2. Map Integration:**

* Use Leaflet.js to integrate the map into the web application.
* Implement event handlers to allow users to select start and end locations by clicking on the map.
* Display markers on the map to indicate the selected start and end points.

**3. Dijkstra's Algorithm Implementation:**

* Design and implement a function to represent the road network as a weighted graph with non-negative edge weights.
* Implement Dijkstra's algorithm to calculate the shortest path between the selected start and end points.

**4. Real-time Traffic Simulation:**

* Obtain real-time traffic data from a traffic data provider (if available).
* Simulate traffic conditions and dynamically adjust edge weights in the graph accordingly to represent traffic congestion.

**5. Route Visualization:**

* Highlight the calculated shortest path on the map using a polyline to provide a visual representation of the selected route.

**6. User Interface Design:**

* Create a user-friendly and intuitive interface for the web application.
* Design interactive controls for users to reset the map, find the shortest path, and create graph visualizations.

**7. Firebase Integration:**

* Set up a Firebase account and create a new Firebase project.
* Use Firebase Hosting to deploy the web application to a publicly accessible URL.
* Configure the necessary Firebase settings to enable hosting for the web page.

**8. Testing and Optimization:**

* Test the application thoroughly for functionality and usability.
* Optimize the code and resources to ensure a smooth and efficient user experience.
* Conduct performance testing to ensure the application performs well under various conditions.

**9. Documentation and Deployment:**

* Prepare detailed documentation explaining the project's architecture, code structure, and usage instructions.
* Deploy the web application to Firebase Hosting to make it accessible to users online.

**Hosting the Web Page using Firebase:**

Firebase Hosting is a cloud-based hosting service provided by Google's Firebase platform. It allows users to deploy static web pages quickly and easily. Here's the process to host the GPS Navigator web page using Firebase:

**1. Install Firebase CLI:**

* Install Firebase Command Line Interface (CLI) using Node Package Manager (NPM) by running the following command in the terminal:

npm install -g firebase-tools

**2. Firebase Login:**

* Log in to your Firebase account by running the following command in the terminal:

**firebase login**

**3. Initialize Firebase Project:**

* Navigate to the project folder in the terminal and run the following command to initialize the Firebase project:

**firebase init**

* Select "Hosting" as the Firebase feature to set up.

**4. Deploy to Firebase Hosting:**

* After initializing, build the project to create the necessary files by running:

**npm run build**

* Deploy the project to Firebase Hosting with the following command:

**firebase deploy**

**5. Access the Web Page:**

* After successful deployment, Firebase will provide a publicly accessible URL where the GPS Navigator web page can be accessed.

##### **DESIGN**

**Design Interface:**

**User Interface (UI) Design:**

The UI features an interactive map interface where users can visually select start and end locations by clicking on the map. Clear buttons for actions such as” Reset Map”,” Find Shortest Path” and “Create Graph Visualization” are included for user interaction.

**Map Integration**:

Utilizes Leaflet.js library to seamlessly integrate the map into the web application.

Displays markers on the map to indicate the selected start and end points chosen by the user.

**Dijkstra's Algorithm Implementation:**

Represents the road network as a weighted graph with nodes representing locations and edges representing road segments with distances (weights).

Applies Dijkstra's algorithm to calculate the shortest path between the selected start and end points.

**Real-time Traffic Simulation:**

Incorporates a traffic data provider (if available) to simulate real-time traffic conditions.

Dynamically adjusts edge weights in the graph to represent traffic congestion and provide accurate travel time estimations.

**Route Visualization:**

Visualizes the calculated shortest path on the map by highlighting the route with a polyline.

Clearly distinguishes the selected route from other road segments for enhanced user understanding.

**Graph Visualization:**

Allows users to generate graph visualizations of the road network, showing nodes and edges.

Displays traffic details on the graph, providing a comprehensive overview of the road network's structure.

**Firebase Hosting:**

Deploys the web application using Firebase Hosting, ensuring fast and reliable access for users.

Provides a publicly accessible URL where users can access the GPS Navigator application.

**Workflow:**

**User Interaction:**

Users access the web application and interact with the intuitive map interface.

They click on the map to select start and end locations for route planning.

**Dijkstra's Algorithm Calculation:**

The selected locations are used as inputs for Dijkstra's algorithm.

The algorithm calculates the shortest path considering distances and edge weights in the graph.

**Real-time Traffic Simulation:**

If enabled, the application simulates real-time traffic conditions to provide estimated travel times.

The graph's edge weights are adjusted dynamically based on traffic data.

**Route Visualization and Presentation:**

The calculated shortest path is highlighted on the map with a polyline, ensuring a clear visualization.

Users can view the selected route, distances, and estimated travel times.

**Graph Visualization:**

Users can choose to generate a graph visualization of the road network, displaying nodes and edges along with traffic details.

**Firebase Hosting and Deployment:**

The web application is deployed using Firebase Hosting, providing a publicly accessible URL.

Users can access the GPS Navigator application from any device with an internet connection.

##### **IMPLEMENTATION**

**Source Code:**

<!DOCTYPE html>

<html>

<head>

    <title>GPS Navigator</title>

    <style>

        body{

            background-color: orange;

        }

        #map {

            height: 450px;

        }

        .controls {

            margin-top: 10px;

        }

        .dark-popup {

            background-color: #f50101;

            color: #fff;

            padding: 10px;

        }

    </style>

    <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />

</head>

<body>

    <h1>GPS Navigator</h1>

    <div id="map"></div>

    <div class="controls">

        <button onclick="resetMap()">Reset Map</button>

        <button onclick="findShortestPath()">Find Shortest Path</button>

        <button onclick="createGraphVisualization()">Create Graph Visualization</button>

    </div>

    <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>

    <script src="https://unpkg.com/leaflet-routing-machine/dist/leaflet-routing-machine.js"></script>

    <script>

        var map;

        var startMarker = null;

        var endMarker = null;

        var routingControl = null;

        var graph = {};

        function initMap() {

            map = L.map('map').setView([0, 0], 1);

            L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

                attribution: '&copy; OpenStreetMap contributors'

            }).addTo(map);

            map.on('click', function(*event*) {

                handleMapClick(*event*.latlng);

            });

        }

        function resetMap() {

            map.setView([0, 0], 1);

            map.removeLayer(startMarker);

            map.removeLayer(endMarker);

            map.removeControl(routingControl);

            startMarker = null;

            endMarker = null;

            routingControl = null;

            graph = {};

        }

        function handleMapClick(*latlng*) {

            if (!startMarker) {

                startMarker = L.marker(*latlng*).addTo(map).bindPopup('Start Location').openPopup();

                console.log('Start Location:', *latlng*.lat, *latlng*.lng);

                map.flyTo(*latlng*, 14);

            } else if (!endMarker) {

                endMarker = L.marker(*latlng*).addTo(map).bindPopup('End Location').openPopup();

                console.log('End Location:', *latlng*.lat, *latlng*.lng);

                findShortestPath();

            }

        }

        function findShortestPath() {

            var start = startMarker.getLatLng();

            var end = endMarker.getLatLng();

            console.log('Finding shortest path between:', start, end);

            if (routingControl) {

                map.removeLayer(routingControl);

            }

            routingControl = L.Routing.control({

                waypoints: [

                    L.latLng(start.lat, start.lng),

                    L.latLng(end.lat, end.lng)

                ],

                routeWhileDragging: false

            }).addTo(map);

            routingControl.on('routesfound', function(*e*) {

                var routes = *e*.routes;

                if (routes && routes.length > 0) {

                    var route = routes[0];

                    var coordinates = route.coordinates;

                    graph['path'] = coordinates;

                    console.log('Path:', coordinates);

*// Simulate traffic details*

                    var trafficDetails = "Traffic is moderate. Estimated time: 15 minutes.";

                    graph['traffic'] = trafficDetails;

                }

            });

        }

        function createGraphVisualization() {

            if (graph && graph.path) {

                var pathCoordinates = graph.path;

*// Save the graph data in sessionStorage to pass it to graph.html*

                sessionStorage.setItem('graphData', JSON.stringify({

                    path: pathCoordinates,

                    traffic: graph.traffic || "No traffic information available."

                }));

*// Open the graph.html in a new window*

                var popupWindow = window.open("graph.html", "\_blank");

            }

        }

        initMap();

    </script>

</body>

</html>

**The Graph visualization**

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">

    <title>Graph Visualization with Traffic Details</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: orange; }

        #map {

            height: 600px;

        }

        .dark-popup {

            background-color: #333;

            color: #fff;

            padding: 10px;

        }

    </style>

    <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css">

</head>

<body>

    <h1>Graph Visualization with Traffic Details</h1>

    <div id="map"></div>

    <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>

    <script>

        function initMap() {

            var map = L.map('map').setView([0, 0], 1);

            L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

                attribution: '&copy; OpenStreetMap contributors'

            }).addTo(map);

*// Get the graph data from sessionStorage*

            var graphData = JSON.parse(sessionStorage.getItem('graphData'));

            if (graphData && graphData.path) {

                var pathCoordinates = graphData.path;

*// Add a start point marker with a custom icon*

                var startPoint = pathCoordinates[0];

                var startIcon = L.icon({

                    iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master/img/marker-icon-green.png',

                    iconSize: [25, 41],

                    iconAnchor: [12, 41],

                    popupAnchor: [1, -34],

            });

                var startMarker = L.marker(startPoint, { icon: startIcon }).addTo(map);

                startMarker.bindPopup('Start Point').openPopup();

*// Add an end point marker*

                var endPoint = pathCoordinates[pathCoordinates.length - 1];

                var endMarker = L.marker(endPoint).addTo(map);

                endMarker.bindPopup('End Point').openPopup();

*// Create a polyline to represent the graph*

                var polyline = L.polyline([], { color: 'purple' }).addTo(map);

                var index = 0;

                var animationInterval = setInterval(function() {

                    if (index >= pathCoordinates.length) {

                        clearInterval(animationInterval);

                        map.fitBounds(polyline.getBounds());

                        return;

                    }

*// Animate the graph drawing*

                    polyline.addLatLng(pathCoordinates[index]);

                    map.panTo(pathCoordinates[index]);

                    index++;

                }, 30); *// Adjust the animation speed here (lower value means faster)*

*// Show traffic details as a popup*

                var trafficDetails = graphData.traffic;

                if (trafficDetails) {

                    var trafficPopup = L.popup({ className: 'dark-popup' })

                        .setLatLng([0, 0])

                        .setContent(trafficDetails)

                        .openOn(map);

                }

            }

        }

        initMap();

    </script>

</body>

</html>

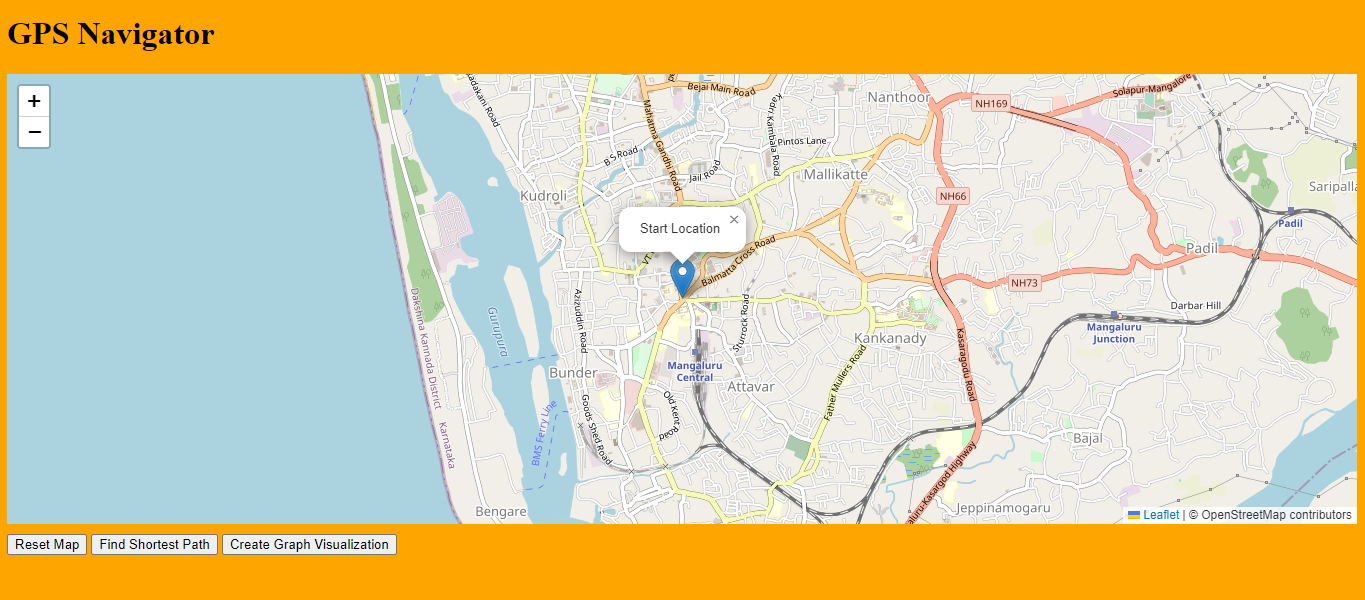
##### **RESULT**

**Outputs:**

****

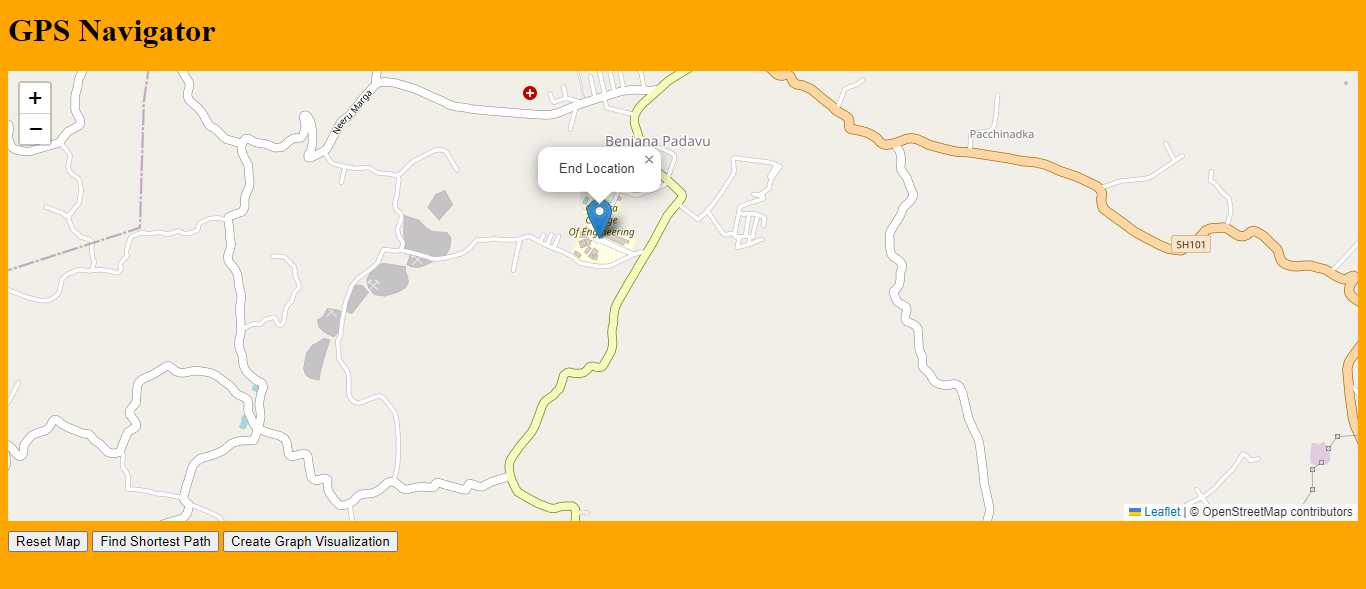
**Figure 1:- Main Window**

Figure 1 provides the starting main window of the web page. Since it won’t relocate or fetch to our current location because we have not used any API’s keys. For Api’s it will cost nearly 300$.



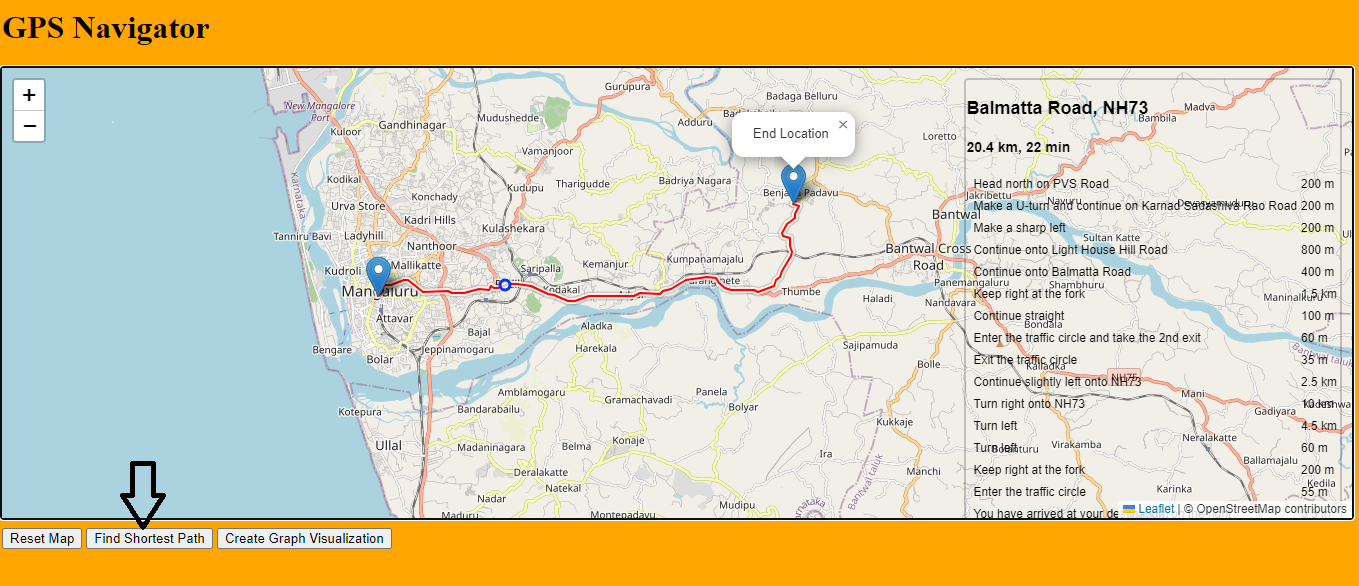
**Figure 2:- Selecting Starting Location**

We need to zoom on the map and we need to choose the starting location from the map.

****

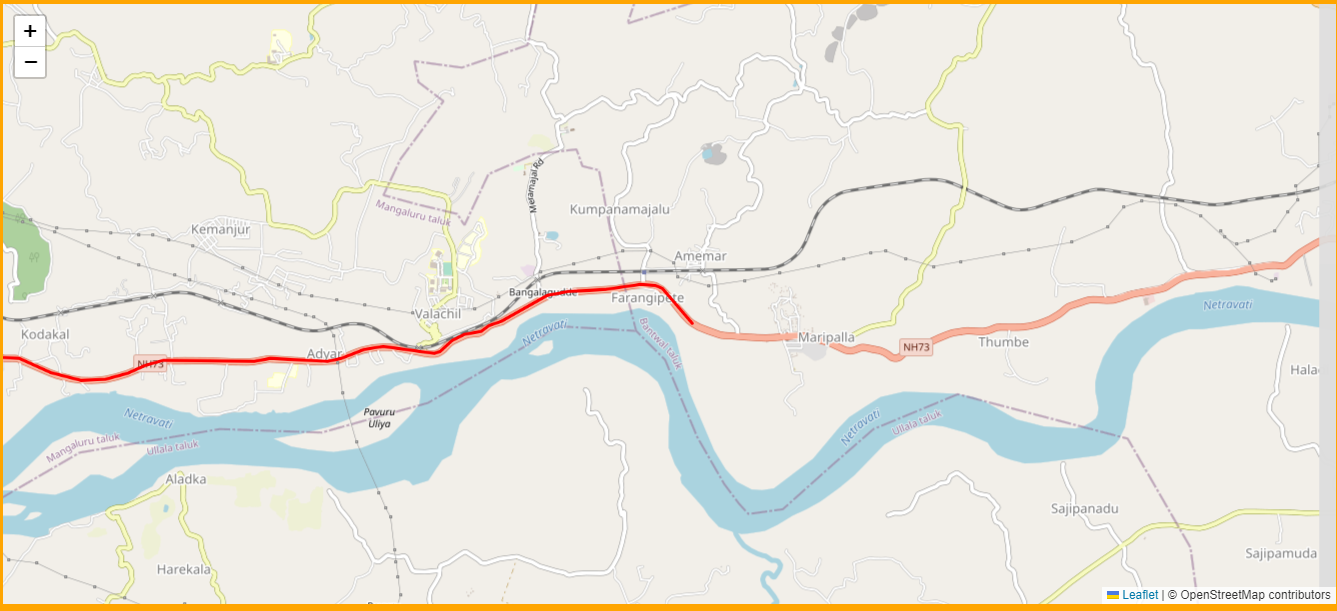
**Figure 3:- Selecting Ending Location**

We need to choose the ending location on the map by clicking.



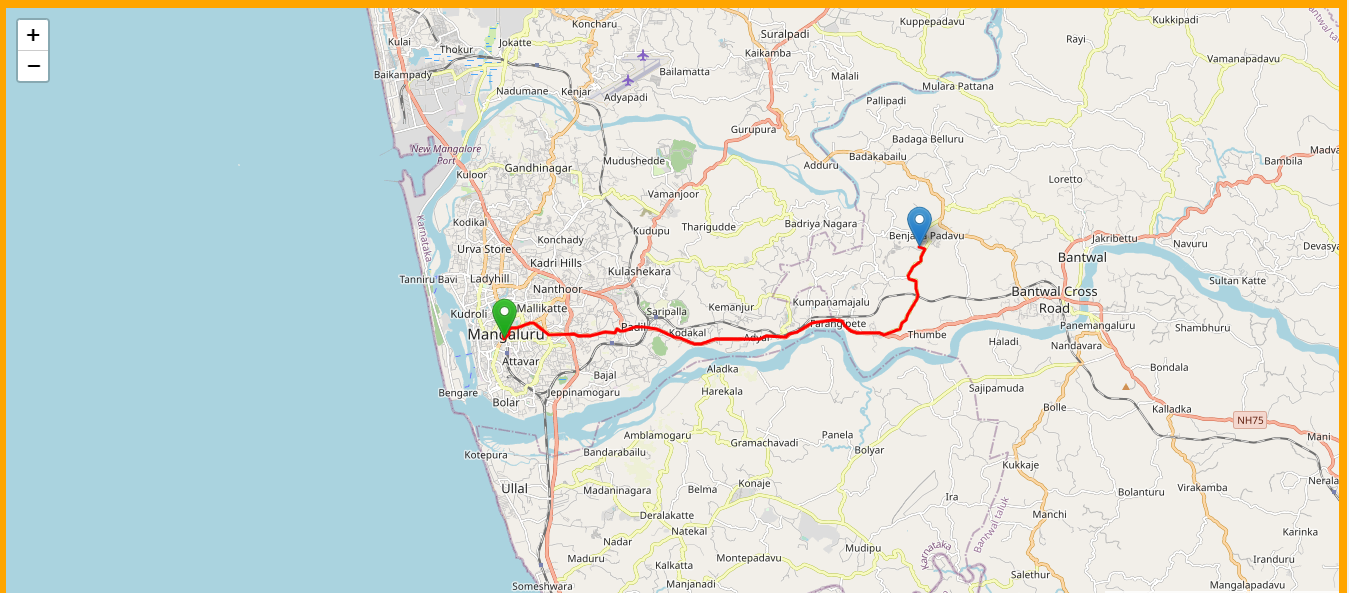
**Figure 4:- Shortest path**

Upon clicking “Find Shortest Path” button it will show the shortest path with a red polyline.



**Figure 5:- Graphical representation**

Upon clicking “Create Graph Visualization” it will redirect to another page and a red polyline will start to move from starting location to ending location. The movement of the line is varying based upon the traffic condition.



**Figure 6:- Final Graphical Representation**

# CONCLUSION

The GPS Navigator project offers a comprehensive and user-friendly solution for efficient route planning and navigation. By combining advanced algorithms, real-time traffic simulation, and interactive map interfaces, the project aims to enhance the overall navigation experience for users. The key highlights and outcomes of the project can be summarized as follows:

**Key Highlights:**

**User-Centric Design:** The project prioritizes user experience with an intuitive map interface and interactive controls, allowing users to easily select start and end locations for route planning.

**Efficient Routing:** Dijkstra's algorithm is leveraged to calculate the shortest path between selected locations, ensuring users receive optimal and time-efficient routes.

**Real-Time Traffic Simulation:** The project simulates real-time traffic conditions to provide users with accurate travel time estimations and dynamic adjustments to route planning based on traffic congestion.

**Visual Route Presentation:** Users can visualize the selected route through map polylines, facilitating a clear understanding of the path they will be taking.

**Graph Visualization:** The option to generate graph visualizations provides users with a deeper insight into the road network's structure, enhancing their understanding of available routes.

**Firebase Hosting:** The project is seamlessly deployed using Firebase Hosting, ensuring widespread accessibility and reliability for users.

**Expected Outcomes:**

**Enhanced Route Planning:** Users will benefit from efficient route planning, leading to reduced travel times and improved navigation.

**Informed Decisions:** Real-time traffic simulation equips users with crucial traffic data, enabling them to make informed decisions and adapt routes as needed.

**Visual Understanding:** The interactive map interface and graph visualizations empower users to visually comprehend their routes, fostering better decision-making.

**Convenient Accessibility:** The deployment on Firebase Hosting makes the application readily accessible to users from various devices and locations.

**Educational Tool:** The project's graph visualization component can serve as an educational tool, helping users understand the underlying road network and traffic patterns.

In essence, the GPS Navigator project aims to revolutionize traditional navigation methods by integrating advanced algorithms, real-time data, and user-friendly interfaces. By providing accurate route planning, traffic insights, and visual aids, the project empowers users to navigate more efficiently and make informed travel decisions. Through these efforts, the GPS Navigator project contributes to enhancing the overall travel experience and promoting effective route optimization.