IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

ApnaRoute: Smart Route Finder and Travel Cost Calculator

Vrushali Rajput[1], Prof.Manoj V. Nikum[2]

MCA, Shri Jaykumar Rawal Institute of Technology, Dondaicha, India[1]

HOD MCA & Assistant Professor, Shri Jaykumar Rawal Institute of Technology, Dondaicha, India[2]

Abstract: ApnaRoute is a web-based route-planning application designed to bring route search, distance and travel-time estimation, fuel-cost calculation and nearby-service discovery into a single interface. Users enter an origin and destination, which are geocoded using Photon with Nominatim as a fallback, and view the locations and route on an interactive OpenStreetMap-based map. The application can present up to three route alternatives—Fastest, Balanced and Scenic—with distance, estimated duration and fuel-cost information. Fuel requirements and estimated cost are calculated from the route distance, user-entered vehicle mileage and fuel price. The system also supports live device location, nearby petrol pumps, hospitals and restaurants, and private saved routes for registered users. The implementation uses HTML, CSS, Bootstrap and JavaScript, Python Flask, Leaflet, OpenStreetMap services and Supabase PostgreSQL with Row Level Security. A sample route test from Kadamwadi, Kolhapur, to Baramati returned 219.3 km and an estimated travel time of 3 hours 7 minutes for the Fastest route; the reported fuel estimate was ■1,412 under the calculator inputs used during that test. The project demonstrates an integrated, open-data-based travel-planning prototype. Travel time is an estimate and does not include live traffic.

Keywords: Route Planning, Fuel Cost Calculator, OpenStreetMap, Leaflet, Flask, Supabase, Geocoding, Nearby Places, Row Level Security.

I. INTRODUCTION

Road journeys between cities, towns and villages require travellers to consider the route, distance, expected travel time, fuel requirement and availability of essential services. These details are often obtained from separate applications or websites, making journey planning less convenient.

ApnaRoute (meaning “my own route”) is a web application that combines these activities in one interface. It provides an interactive map, up to three alternative routes, a fuel-cost calculator and a nearby-places finder. Registered users can save routes and access them later.

The main objective is to develop a simple and responsive travel-planning system that accepts Indian locations, compares available route options, estimates fuel cost using the user's vehicle details, and identifies nearby petrol pumps, hospitals and restaurants. The system uses live responses from open geographic services rather than hard-coded route distances.

II. LITERATURE REVIEW

Web mapping applications combine map tiles with geographic overlays to display locations and routes in a browser. OpenStreetMap (OSM) provides openly available map data, while Leaflet offers a lightweight JavaScript library for interactive maps [1], [2]. These tools provide a practical foundation for low-cost mapping applications.

Route planning requires converting place names into geographic coordinates and finding a path through a road network. Photon and Nominatim provide geocoding services based on OpenStreetMap data [3], [4]. The Open Source Routing Machine (OSRM) provides route geometry, distance and duration through an HTTP API [5].

Nearby facilities can be retrieved from OpenStreetMap data using the Overpass API [6]. Flask supports the server-side application workflow [7], while Supabase provides PostgreSQL storage and authentication facilities, including Row Level Security for restricting access to user-owned records [8]. ApnaRoute combines these services to address the gap between route display, trip budgeting and nearby-service search.

III. METHODOLOGY

A. System Architecture and Application Workflow

ApnaRoute follows a client–server architecture with three logical layers. The presentation layer uses HTML, CSS, Bootstrap and JavaScript. Leaflet displays the map and route overlays using OpenStreetMap tiles. The application layer uses Python Flask to validate requests, call external services, calculate fuel estimates and return results. The data layer uses Supabase PostgreSQL and Supabase Authentication to store profiles, saved routes, preferences and saved

calculations.

Place names are sent to Photon for geocoding. If no result is returned, Nominatim is used as a fallback. Once origin and destination coordinates are available, Flask requests route information from an OSRM-based routing service. The application displays up to three alternatives, labelled Fastest, Balanced and Scenic, when alternatives are available. Distance in metres is converted to kilometres and duration in seconds is formatted as hours and minutes.

B. Fuel Cost Calculation and Nearby Services

The fuel calculator uses the following formulas:

Fuel Required	=	Distance ÷ Mileage
Fuel Cost	=	Fuel Required × Fuel Price per Unit

For fuel estimation, the user selects a vehicle type and fuel type, then enters mileage and fuel price. The system validates that mileage and price are positive values before calculating. Nearby-place requests use the Overpass API to search for petrol pumps, hospitals and restaurants around the selected origin. Results are shown as map markers and a list; selecting a place draws a path from the origin to that place.

Registered users can save routes and view them again. Supabase Row Level Security policies restrict records to their owners. Secret keys and database credentials are stored in environment variables and are not exposed in browser code. The project report identifies the main application modules as login and registration, dashboard, route finder, route details, fuel calculator, nearby places and saved routes.

IV. RESULT AND DISCUSSION

The project report describes a sample journey from Kadamwadi, Kolhapur, Karvir, Maharashtra, to Baramati, Maharashtra. The Fastest route was marked as recommended and returned a distance of 219.3 km with an estimated duration of 3 hours 7 minutes. The reported fuel-cost estimate was ■1,412; this value depends on the mileage and fuel price entered during the test.

Functional checks recorded as Pass in the report include valid route search, distance and time display, fuel-cost display, and loading of the home page and information sections. Other listed checks—including empty or unknown locations, invalid calculator inputs, live GPS, nearby-place selection, registration, saved-route reuse, privacy isolation and API failure handling—are marked for verification in the project report. They are therefore not presented here as confirmed successful tests.

The implementation depends on public services whose response time, coverage and usage limits may vary. Route duration does not include live traffic, and alternative routes may not always be available. The fuel estimate is based on user-provided mileage and price, so it is an approximation rather than a guaranteed trip expense.

TABLE I. SAMPLE ROUTE RESULT

Route Type	Distance	Estimated Time	Estimated Fuel Cost
Fastest (Recommended)	219.3 km	3 hr 7 min	■ 1,412

V. CONCLUSION

ApnaRoute brings route search, alternative-route comparison, distance and time estimation, fuel-cost calculation, nearby-service discovery and saved routes into a single web application. By combining Flask, Leaflet, OpenStreetMap-based services and Supabase, the project demonstrates a practical, low-cost travel-planning prototype for Indian users. The sample route result illustrates the integration of routing data with the fuel calculator. Further validation of the test cases marked for verification is required before making broader reliability claims. Future enhancements identified in the project report include live traffic, toll estimation, multi-stop journeys, automatic fuel-price updates, turn-by-turn and voice navigation, offline support, a mobile application, additional nearby-place categories, multilingual support and route sharing or PDF export.

ACKNOWLEDGMENT

The author expresses sincere gratitude to **Mr. Manoj V. Nikum**, HOD MCA & Assistant Professor, Shri Jaykumar Rawal Institute of Technology, Dondaicha, for his guidance and support. The author also thanks the faculty members and the open-source communities behind OpenStreetMap, Leaflet, Flask, Bootstrap and Supabase for the tools and documentation that supported this project.

REFERENCES

[1] OpenStreetMap Foundation, "OpenStreetMap." [Online]. Available: https://www.openstreetmap.org
[2] V. Agafonkin, "Leaflet – an open-source JavaScript library for interactive maps." [Online]. Available: https://leafletjs.com
[3] Komoot, "Photon – open-source geocoder for OpenStreetMap data." [Online]. Available: https://photon.komoot.io
[4] Nominatim, "Nominatim documentation – geocoding with OpenStreetMap data." [Online]. Available: https://nominatim.org
[5] Project OSRM, "Open Source Routing Machine – HTTP API documentation." [Online]. Available: https://project-osrm.org
[6] Overpass API, "Overpass API user's manual." [Online]. Available: https://wiki.openstreetmap.org/wiki/Overpass_API
[7] Pallets Projects, "Flask documentation." [Online]. Available: https://flask.palletsprojects.com
[8] Supabase Inc., "Supabase documentation – Auth and Row Level Security." [Online]. Available: https://supabase.com/docs
[9] The PostgreSQL Global Development Group, "PostgreSQL documentation." [Online]. Available: https://www.postgresql.org/docs
[10] Bootstrap Team, "Bootstrap documentation." [Online]. Available: https://getbootstrap.com/docs