Project Report

1. Project Title

Smart City Navigator

2. Objective

The primary objective of this project is to develop an interactive Smart City Navigation System that assists users in finding the best route between two city locations. The system applies graph theory and shortest-path algorithms through the OpenRouteService (ORS) API, integrating real-time geocoding, routing, and visualization for educational and practical navigation purposes.

3. Expected Outcomes

• Understand how graph-based algorithms (like Dijkstra and A*) are applied in real-world routing systems.

• Learn how to integrate APIs for geocoding, autocomplete, and route optimization.

• Gain hands-on experience in building a data-driven, interactive web application using Python and Streamlit.

• Be able to visualize and interpret route data and navigation instructions dynamically.

4. Tools & Technologies

• Programming Language: Python

• Framework: Streamlit

• APIs: OpenRouteService (ORS) – for geocoding, autocomplete, and routing

• Visualization Library: Folium (Leaflet.js)

• Fuzzy Matching Library: rapidfuzz

• Platform: Localhost / Web (Streamlit App)

• Data Source: OpenStreetMap via OpenRouteService

5. Team Information

• Group No.: 01

• Team Members:

1. Rehan Samnani(24102C0039)
2. Jay Keluskar(24102C0086)
3. Yash Waghmare(24102C0031)
4. Rishi Shah(24102C0075)
5. Arnav Mundada (24102C0024)

6. Methodology / Procedure

1. Designed a Streamlit-based user interface for location input and configuration.

2. Integrated OpenRouteService APIs to perform geocoding, autocomplete, and route generation.

3. Applied fuzzy matching (rapidfuzz) to enhance user input handling and location search accuracy.

4. Modeled the city road network as a graph (nodes = intersections, edges = roads).

5. Utilized shortest-path algorithms (Dijkstra / A*) through ORS for optimal routing.

6. Visualized the computed routes on an interactive map using Folium.

7. Implemented a GeoJSON export feature for route data.

8. Validated the application's performance and user experience through testing and demonstrations.

7. Findings

• The autocomplete and fuzzy search ensured seamless location entry even with spelling errors.

• Graph-based routing provided accurate, efficient, and alternative route suggestions.

• Visualization via Folium enhanced user understanding of route data and navigation.

• The downloadable GeoJSON feature enabled further GIS analysis.

• The integration of multiple APIs and Python libraries proved the flexibility of modular web app design.

8. Conclusion

The Smart City Navigator successfully demonstrates how graph theory and modern APIs can be combined to create a real-time, intelligent navigation system. The project bridged theoretical knowledge with practical implementation, giving students experience in data structures, algorithms, and full-stack Python development. This solution can be extended for real-world smart city applications, including traffic management, logistics routing, and emergency response planning.