Software Requirements Specification

For

Travel Optimizer: A comparative to find out the optimal path

<Date>

Prepared by

Specialization	SAP ID	Name
Data Science	500106020	Isha
Data Science	500107837	Jahanvi Sethi
Data Science	500102009	Saumya Gupta

UPES
UNIVERSITY WITH A PURPOSE

Department of Informatics
School Of Computer Science
UNIVERSITY OF PETROLEUM & ENERGY STUDIES,
DEHRADUN- 248007. Uttarakhand

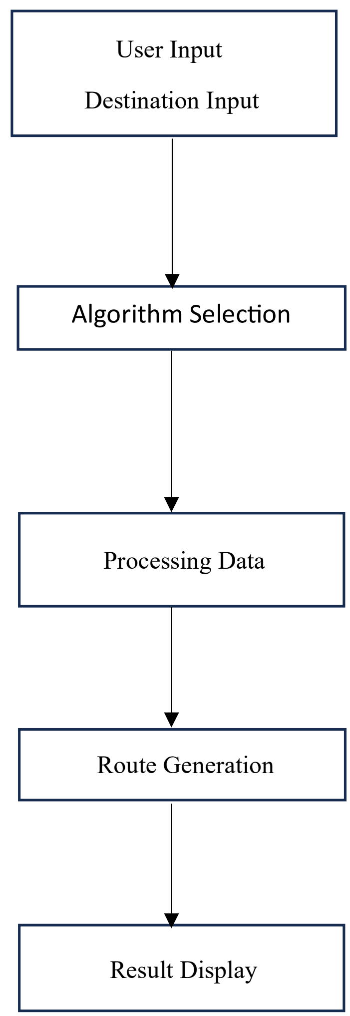
Table of Contents

Revision History

Date	Change	Reason for Changes	Mentor Signature

1	INTRODUCTION	
	<u>1.1 Purpose of the Project</u>	The purpose of this project is to develop a Travel Optimizer that provides users with the most efficient route between multiple locations. This will minimize travel time, distance, or cost, tailored to various real-world travel scenarios. The system compares different pathfinding algorithms and selects the most suitable for given travel conditions. With the growing complexity of travel needs and the increased availability of real-time data, existing solutions often fall short in handling large-scale scenarios, where multiple destinations and constraints are involved. This project addresses those gaps by implementing a comparative study of different pathfinding algorithms, to find the best approach based on specific travel conditions.
	<u>1.2 Target Beneficiary</u>	**1.Travelers: Planning Routes for Multi-Destination Trips:** The Travel Optimizer helps travellers efficiently plan multi-destination routes by recommending the best sequence to minimize time, distance, or cost. Travelers can adjust routes on the go to account for traffic, or personal detours, making it ideal for tourists and business travellers needing flexible, hassle-free planning. **2.Businesses: Logistics and Delivery Services Requiring Optimized Route Planning:** The Travel Optimizer enables logistics and delivery companies to reduce costs, improve efficiency, and ensure timely deliveries by optimizing routes for fuel, labour, and time savings. It adapts to real-time conditions, offering alternative routes for delays, and supports custom constraints like vehicle capacity and delivery deadlines to maximize productivity. **3.Researchers and Developers: Academic and Development Purposes in Optimization Algorithms:** The Travel Optimizer offers researchers and developers a robust platform for testing and comparing algorithms, to assess performance in real-world scenarios. It supports studies in optimization and AI, while providing developers with a customizable environment to implement new algorithms and address specific optimization challenges.
	<u>1.3 Project Scope</u>	The project will implement and compare pathfinding algorithms, to find optimal routes between multiple cities. The system will address real-world constraints such as travel costs, time fluctuations due to traffic, and specific user restrictions, making it a practical tool for diverse travel scenarios. Additionally, there is potential to extend the system to include a variety of other algorithms, enhancing its adaptability for personal travel, business logistics, and academic research.
	<u>1.4 References</u>	1. T. Huang, Y.-J. Gong, S. Kwong, H. Wang, and J. Zhang, "A niching memetic algorithm for multi-solution traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 24, no. 3, pp. 508-522, Jun. 2020. 2. Y. Liu et al., "Multi-modal multi-objective traveling salesman problem and its evolutionary optimizer," *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, pp. 770-777, 2021.

2	PROJECT DESCRIPTION	
	2.1 Reference Algorithm	State the reference algorithm for the project and identify the required data structure (**Mandatory for Minor1**) Or/Add design algorithm justifying the methodology of the project
	2.2 Characteristic of Data	**Graph**: Represents locations as nodes and paths as edges, with weighted attributes like time, distance, or cost. **Priority Queue**: Essential for selecting nodes efficiently in Dijkstra's Algorithm. **HashMap**: For quick access to node-related data. **Arrays and Lists**: For maintaining sequences of distances, predecessors, and visited nodes. **Set**: To track visited nodes.
	2.3 SWOT Analysis	**Strengths:** • **Efficiency:** Utilizes advanced algorithms for quick, accurate route recommendations that save time and costs. • **Customization:** Allows users to input several locations for tailored travel solutions. **Weaknesses:** • **Algorithm Limitations:** May not always deliver optimal results in dynamic environments with complex constraints. • **Data Dependency:** Relies on the quality of underlying data (maps, traffic updates), which can vary regionally. • **Development Complexity:** Maintaining multiple algorithms and ensuring integration can complicate development and increase costs. **Opportunities:** • **Market Expansion:** Increasing demand for efficient travel solutions in logistics and tourism offers growth potential. • **Technological Integration:** Incorporating AI and machine learning for smarter route planning based on historical data. • **Partnerships:** Collaborating with mapping services and transportation agencies to enhance data accuracy and capabilities. **Threats:** • **Competition:** Established route planning apps and startups create a competitive landscape requiring constant innovation. • **Dependency:** Dependence on external data sources for accuracy.
	2.4 Project Features	**Multi-Algorithm Comparison:** Users can compare routes generated by various pathfinding algorithms to identify the most efficient option. **User-Friendly Interface:** An intuitive interface makes it easy for users to input destinations and view route options. **Route Visualization:** Provides clear visual representations of suggested routes on a map for better understanding. **Performance Metrics:** Displays key metrics such as total distance and estimated travel time.

	2.5 User Classes and Characteristics	**Individual Travelers**: Need intuitive UI and route planning based on personal preferences. **Logistics Managers**: Require route efficiency metrics like cost and time savings. **Developers and Researchers**: Need access to algorithms and performance metrics for comparative studies.
	2.6 Design and Implementation Constraints	**Hardware Limitations**: Performance depends on the computational power available. **Data Quality**: Real-time travel conditions may affect accuracy. **Scalability**: Algorithm performance may decrease with a large number of nodes/edges.
	2.7 Design diagrams	**Flow Diagram:** 
	2.8 Assumption and Dependencies	**Stable Internet Connectivity**: For real-time data fetching the system assumes stable internet connectivity. This is especially important for travellers using the optimizer on the go.

		Standard Input Formats: User inputs (such as destination names or coordinates) are provided in a standard, easily interpretable format, ensuring compatibility with map data and algorithms. **Map and Traffic Data Providers**: The Travel Optimizer depends on third-party providers for accurate map. This may include APIs such as Google Maps, or similar services. **Device Compatibility**: The system's user interface and performance are dependent on compatibility with various web browsers as it is deployed as a web-based solution.
3	SYSTEM REQUIREMENTS	
	<u>3.1 User Interface</u>	**Destination Input Section**: It features an input box that takes the number of destinations, on the basis of that the system takes the number of destinations providing a straightforward way to set up multi-stop routes. **Map Display**: Once the route is calculated, the optimal route is displayed on a map with markers at each destination.
	<u>3.2 Software Interface</u>	**Software Interface Data Integration:** Ability to import and process real-time data from sources like OpenStreetMap. **Algorithm Modules**: Each algorithm is modular, allowing easy updates or additions.
	<u>3.3 Database Interface</u>	**Graph Database**: Stores nodes and edges efficiently to represent locations and paths, allowing for quick retrieval and manipulation of route data. This structure supports complex queries needed for pathfinding and route optimization.
	<u>3.4 Protocols</u>	**HTTP/HTTPS**: For accessing map data and travel conditions. **API Protocols**: To integrate data from external sources.
4	NON-FUNCTIONAL REQUIREMENTS	
	<u>4.1 Performance requirements</u>	**Graph Database**: Stores nodes and edges efficiently to represent locations and paths, allowing for quick retrieval and manipulation of route data. This structure supports complex queries needed for pathfinding and route optimization. **Concurrency**: The system should support multiple users or processes accessing and computing routes simultaneously without performance degradation. This is essential for applications that might serve multiple travellers or delivery vehicles at the same time. **Response Time**: The system should provide a consistent and low response time for user interactions, such as submitting routes or adjusting destination lists. Ideally, feedback should be instantaneous, contributing to a smooth user experience.
	<u>4.2 Security requirements</u>	**Data Privacy**: Secure handling of user data, especially if storing location information. **Data Integrity**: Ensuring the accuracy of route data, preventing unauthorized modifications.
	<u>4.3 Software Quality Attributes</u>	**Reliability**: System should maintain consistent performance across various inputs. **Usability**: User-friendly interface for non-technical users.

		Maintainability: Code should be modular and well-documented for easy updates. **Scalability**: Efficient handling of increasing data volume. **Performance**: The system should deliver high performance, with fast response times for route calculations and data processing. **Extensibility**: The architecture should allow for easy addition of new features or components without significant rework, enabling the system to evolve with user needs and technological advancements.
5	Other Requirements	Define any other requirements not covered elsewhere in the SRS.
Appendix A: Glossary		**Node**: Represents a location or point of interest within the graph, serving as a starting, stopping, or destination point in route calculations. **Edge**: A connection or pathway between two nodes in the graph, which can represent roads, paths, or any traversable route. **Algorithm**: A defined computational method or procedure used to determine the optimal paths between nodes based on specified criteria, such as distance or travel time.
Appendix B: Analysis Model		This section describes the framework used for evaluating and comparing the performance of different pathfinding algorithms implemented within the Travel Optimizer. It includes metrics such as execution time, memory usage, and accuracy of route calculations under various scenarios.
Appendix C: Issues List		This is a dynamic list of the open requirements issues.