2024-2025 Project 3: Call for Tender

Your task is to develop a high-performance software solution that calculates the quickest path between two landmarks in the United States. Below are the project requirements and expectations:

Core Requirements

1. **Programming Language**:
   The software must be implemented in **C++** to achieve optimal performance.
2. **REST API Specification**:
   The software will expose its functionality through a **REST API** running on an HTTP server (localhost). The API will include a single GET endpoint with the following features:
   - Input: IDs of the source and destination landmarks.
   - Output: Travel time and the ordered list of landmarks in the path.
   - Response Formats: Support both **XML** and **JSON** for response payloads.
3. **Data Source**:
   You are provided with a file (USA-roads.csv) containing approximately **24 million nodes** in the format:

   Landmark_A_ID,Landmark_B_ID,Time

   - Each line represents a connection between two landmarks with a travel time (expressed as an integer in an unspecified unit of time).
   - Connections are bidirectional, meaning if a connection exists from A to B, the same applies for B to A (even if not explicitly listed).

Performance Goals

- **Response Time**: The API must handle **all queries within 1 second** on a typical laptop.
- **Approximation Heuristics**: To prioritize speed over precision, your solution may use heuristics. The returned path should not exceed the shortest path duration by more than **10%**.

Data Integrity Verification

Before utilizing the dataset, ensure the data integrity by performing the following checks:

1. **Graph Validation**: Verify that the file forms a **Directed Acyclic Graph** (DAG) and is free of loops.

2. **Connectivity Check**: Ensure that the graph is fully connected, meaning it is possible to navigate between any two landmarks.

Since data integrity checks are performed rather unfrequently, using a relatively naïve, inefficient algorithm is possible. You can also use a different language than C++ if it is more convenient.

Expected Deliverables

1. **C++ Source Code**: Including comments and clear documentation. The code has to be of your own creation, and you should not use libraries beside STL and what is required for the Web server.
2. **Time and Space Complexity**: Big O notation for the main algorithms.
3. **REST API Implementation**: Demonstrating the ability to handle multiple formats (XML and JSON).
4. **Test Suite**: Tests to validate correctness, performance, and compliance with the 10% approximation rule.
5. **Data Validation Tool**: A utility to verify the integrity of the provided CSV file.

This project encourages you to explore and implement efficient algorithms tailored to handle large-scale datasets, while also considering real-world constraints like speed and accuracy.

Note : Functional Specification and User Manual being somewhat equivalent for such a project, Program Manager and Technical Writer will join forces to produce both documents together.