Final Project Proposal

'Journey Guide'

Session: 2023 – 2027

Submitted by:

Saad Tahir 2023-CS-62

Muhammad Talha 2023-CS-70

Supervised by:

Sir Nazeef-ul-Haq

Sir Waseem

Course:

CSC-200L Data Structures and Algorithms Lab

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Contents

Introduction:-

- The Journey Map project aims to provide a solution for planning efficient travel routes between two locations.
- It will use graphs and algorithms to determine the shortest path based on factors like distance, time, and cost.

Objectives:-

- **Graph System:**

  Create a graph that connects locations, making it easy to find routes between them.

- **Shortest Path:**

  Use Dijkstra's algorithm to find the quickest or cheapest route between two points.

- **Route Options:**

  Let users choose between different travel factors like time, cost, or distance for optimal routes.

- **User Input:**

  Allow users to enter starting and ending points to get immediate route suggestions.

- **Simple Interface:**

  Build an easy-to-use interface that shows routes, times, and distances clearly.

Core Features:-

Graph Representation:

- **Adjacency List or Matrix:** Represent cities, stations, or locations as nodes and routes as edges.
- **Weighted Graphs:** Assign weights to edges based on distance, time, or cost.

Algorithms:

- **Shortest Path:** Implement algorithms like Dijkstra's or A* to find the quickest or most efficient routes.
- **Minimum Spanning Tree:** Use Prim's or Kruskal's to optimize connections, such as building a new network.
- **DFS and BFS:** Explore connectivity between destinations and find all possible routes.
- **Backtracking:** Solve complex problems like finding all paths or planning circular routes.

Data Structures:

- **Trees:** Use trees for hierarchical data, such as a train network with hubs and branches.
- **Heaps/Priority Queues:** Optimize computations for the shortest path or cheapest route.
- **Hashmaps/Dictionaries:** Efficiently store and retrieve location details.
- **Stacks and Queues:** Manage stops or ticket bookings in an organized way.

Implementation Plan: Step-by-Step

1. Data Representation

- Use a **graph** to represent the network of locations:
    - **Nodes**: Represent locations (e.g., cities, stations).
    - **Edges**: Represent routes between locations, with weights indicating distance, time, or cost.
- Store the graph data in an **adjacency list** for efficient access and updates.

2. Core Backend Logic

- **Graph Construction**:
    - Develop methods to dynamically add nodes (locations) and edges (routes).
    - Assign weights to edges based on the selected optimization factor (e.g., shortest distance or lowest cost).
- **Algorithms**:
    - Implement **Dijkstra's algorithm** for shortest path calculation.
    - Include additional algorithms like **A*** for heuristic-based optimizations or **Prim's/Kruskal's** for network optimization tasks.
- **Data Structures**:
    - Use **priority queues** for efficient pathfinding.
    - Use **hashmaps** for quick location lookups and route validation.

3. User Interaction

- **Input Handling**:
    - Prompt users to input:
        - Starting location.
        - Destination.
        - Optimization preferences (e.g., time, cost, or distance).
    - Validate user inputs against the graph to ensure the locations exist.
- **Output Processing**:
    - Generate the optimal route using the chosen algorithm.
    - Calculate the total cost, distance, or time for the route.

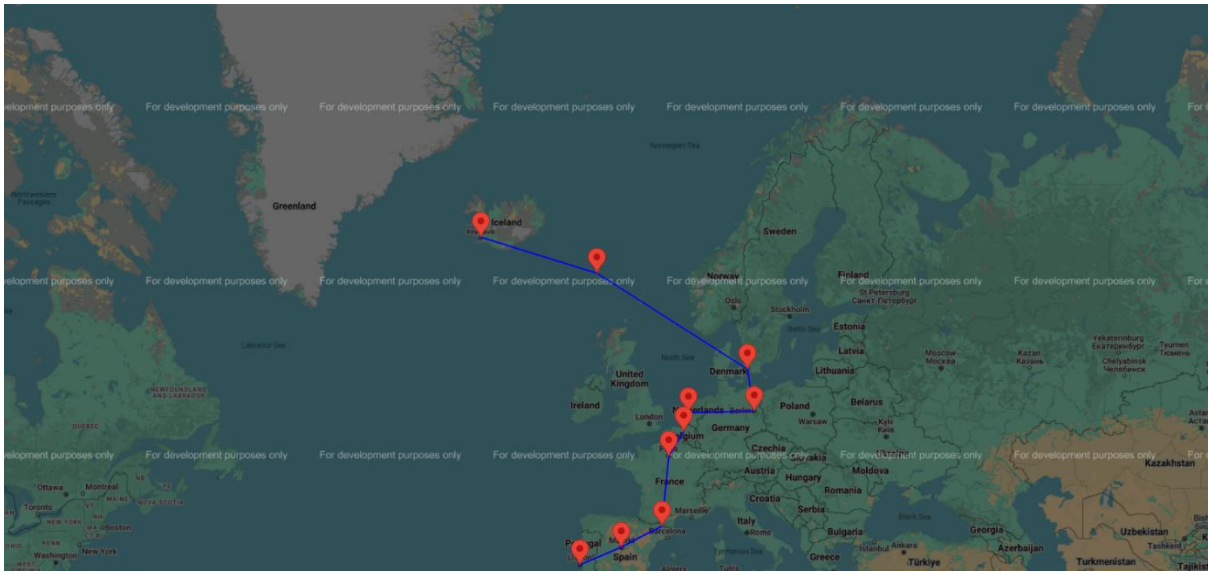
4. Frontend Development

- Design a user-friendly interface:
    - A **text-based UI** can display input fields for locations and output the route details.

- For a more interactive experience, create a **visual representation**:
  - Nodes (locations) displayed as points on a map.
  - Edges (routes) shown as connecting lines with labels for weights.
  - Highlight the calculated path for better clarity.

Visual Representation:-

- Shortest path from Iceland to Portugal.



Assumptions:-

"The project uses mock datasets of cities and routes, assuming all connections are static and distances are accurate. In future, dynamic data from APIs can be integrated."