Vivekanand Education Society's Institute Of Technology

Department Of Information Technology

DSA mini Project

A.Y. 2025-26

Title:

Title: TRAVEL ROUTE PLANNER – INDIA

Sustainability Goal: Encourage local and public transport.

Domain:

DATA STRUCTURES AND ALGORITHMS

Member:

HIMANI RUPLANI

Mentor Name:

MISS. KAJAL JEWANI

1 NO POVERTY

2 ZERO HUNGER

3 GOOD HEALTH AND WELL-BEING

4 QUALITY EDUCATION

5 GENDER EQUALITY

6 CLEAN WATER AND SANITATION

7 AFFORDABLE AND CLEAN ENERGY

8 DECENT WORK AND ECONOMIC GROWTH

9 INDUSTRY, INNOVATION AND INFRASTRUCTURE

10 REDUCED INEQUALITIES

11 SUSTAINABLE CITIES AND COMMUNITIES

THE GLOBAL GOALS

For Sustainable Development

12 RESPONSIBLE CONSUMPTION AND PRODUCTION

13 CLIMATE ACTION

14 LIFE BELOW WATER

15 LIFE ON LAND

16 PEACE AND JUSTICE STRONG INSTITUTIONS

17 PARTNERSHIPS FOR THE GOALS

Content

1. Introduction to the Project
2. Problem Statement
3. Objectives of the Project
4. Scope of the Project
5. Requirements of the System (Hardware, Software)
6. ER Diagram of the Proposed System
7. Data Structure & Concepts Used
8. Algorithm Explanation
9. Time and Space Complexity
10. Front End
11. Implementation
12. Gantt Chart

13. Test Cases
14.Challenges and Solutions
15. Future Scope
16. Code
17. Output Screenshots
18. Conclusion
19. References (in IEEE Format)

Introduction to Project

The project is a Travel Route Planner for India, designed as a Data Structures and Algorithms (DSA) mini-project simulating optimal travel planning across 20 major Indian cities. It enables users to select cities and find the best travel route based on distance, time, or cost using advanced graph algorithms.

Problem Statement

Travelers, logistics firms, and planners often need to discover efficient routes that save them time or cost. Manual route determination is complex, especially for large city networks. The project addresses this by automating route planning with graph algorithms to find the shortest, fastest, or cheapest paths.

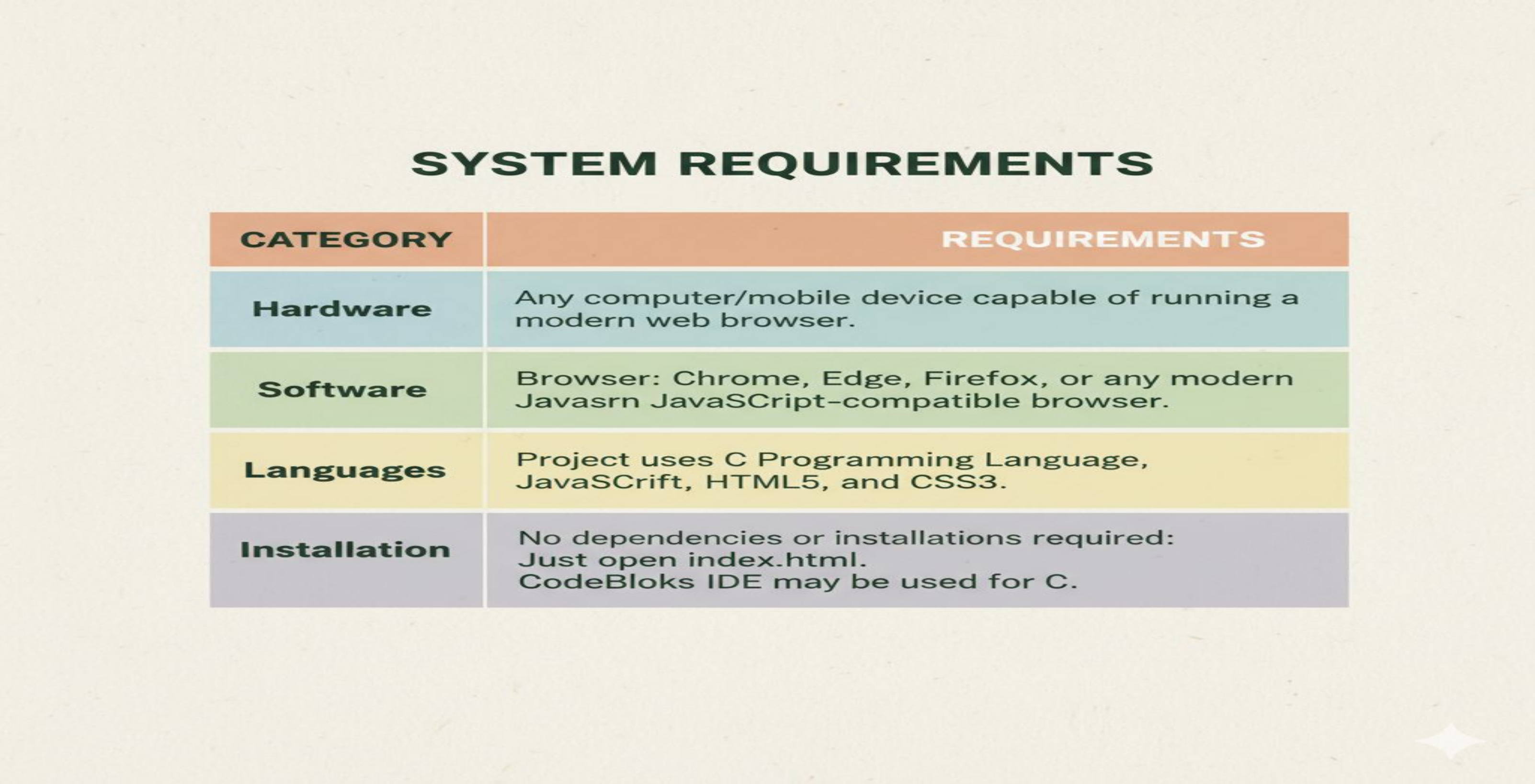
Objectives of the project

- Develop a web application for travel route planning using graph algorithms.
- Allow users to select criteria: shortest distance, fastest route, or lowest cost.
- Provide interactive visualization and detailed step-by-step directions.
- Demonstrate real-world applications of DSA concepts.

Scope of the Project

- 20 major Indian cities and over 50 routes.
- Integrates 4 major pathfinding algorithms: Dijkstra, A*, BFS, DFS.
- Real-world use cases include navigation, logistics, travel planning, and network routing.

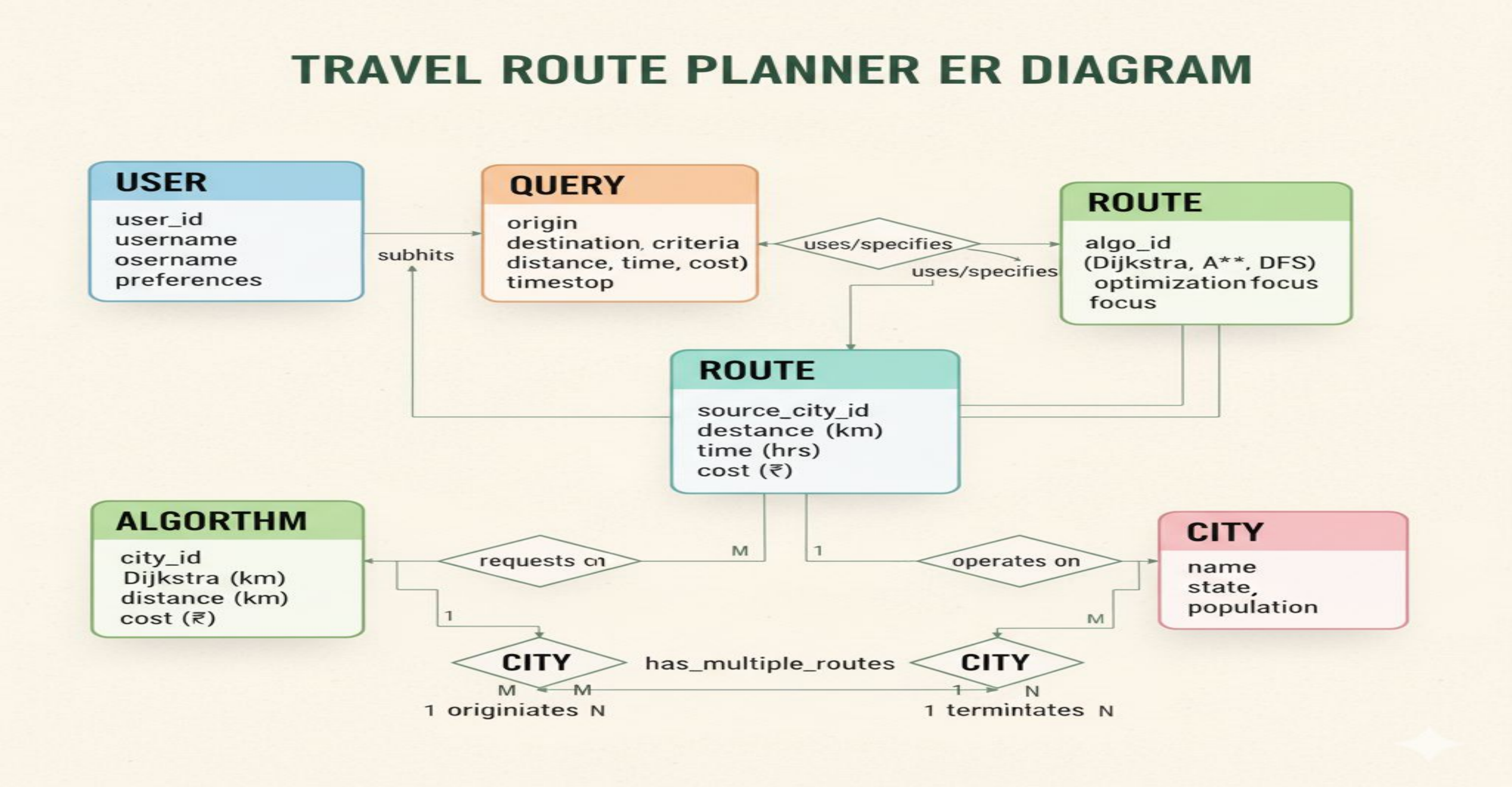
Requirements of the system (Hardware, software)



SYSTEM REQUIREMENTS

CATEGORY	REQUIREMENTS
Hardware	Any computer/mobile device capable of running a modern web browser.
Software	Browser: Chrome, Edge, Firefox, or any modern Javasrn JavaSCript-compatible browser.
Languages	Project uses C Programming Language, JavaSCrift, HTML5, and CSS3.
Installation	No dependencies or installations required: Just open index.html. CodeBloks IDE may be used for C.

ER diagram of the proposed system



Data Structures and Concepts Used

- Graph representation: Cities as nodes (V); routes as weighted edges (E) (distance, time, cost).
- Adjacency list: For efficient graph traversal and storage.
- Queue & Set: Used in BFS/DFS implementations.
- Maps/Objects: For storing city and route information efficiently.

Algorithm Explanation

- Dijkstra’s Algorithm: Finds the shortest path in weighted graphs—guaranteed optimal.
- A* (A-Star): Fast optimal pathfinding using heuristics (straight-line distance) to guide the search.
- BFS (Breadth-First Search): Finds the path with the minimum number of stops (edges/routes).
- DFS (Depth-First Search): Explores all paths, often used to check connectivity, not necessarily the shortest route.

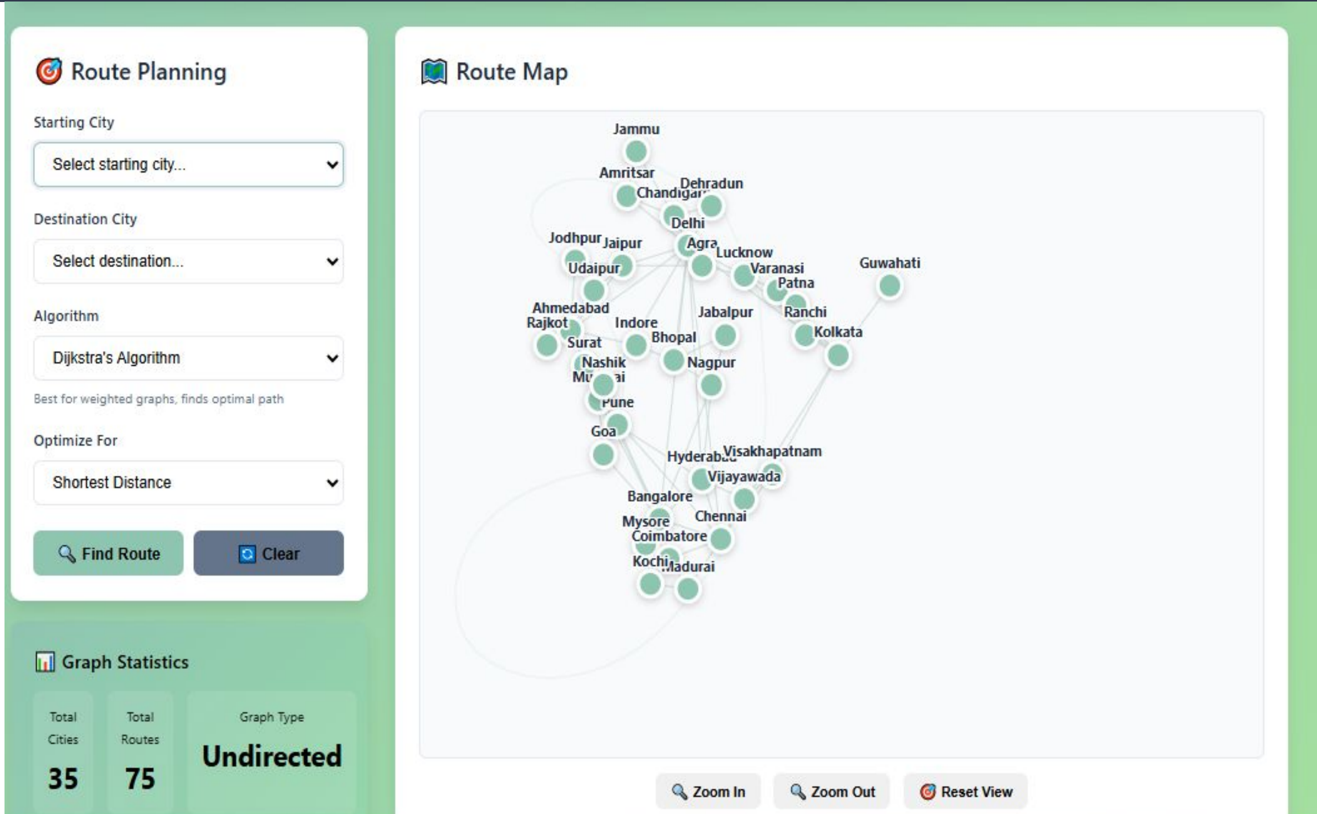
Time And Space Complexity

- Dijkstra: O(E+V\log V)
- A*: Similar to Dijkstra but can often finish faster due to effective heuristics.
- BFS/DFS: O(V+E)

Where: V = number of cities (nodes); E = number of routes (edges).

Front End

- Written in **HTML5**, **CSS3**, **JavaScript**.
- **Controls:** City selection, algorithm selection, optimization criteria (distance/time/cost).
- Features an **interactive visualization** (SVG/Canvas), a statistics panel, and result cards with step-by-step directions.



Route Planning

Starting City: Select starting city...

Destination City: Select destination...

Algorithm: Dijkstra's Algorithm

Best for weighted graphs, finds optimal path

Optimize For: Shortest Distance

Graph Statistics

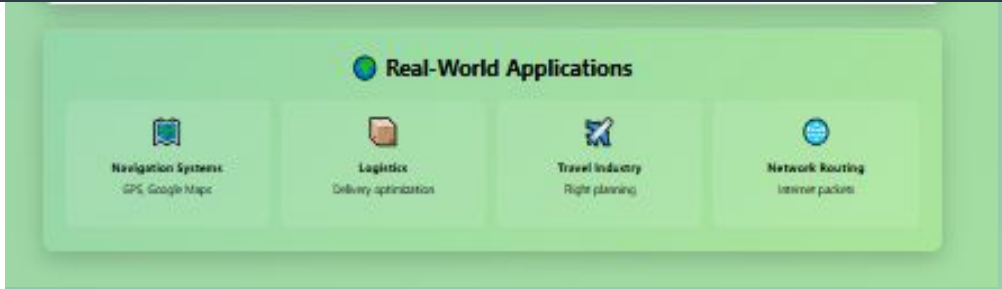
Total Cities: 35

Total Routes: 75

Graph Type: Undirected

Route Map

Front End



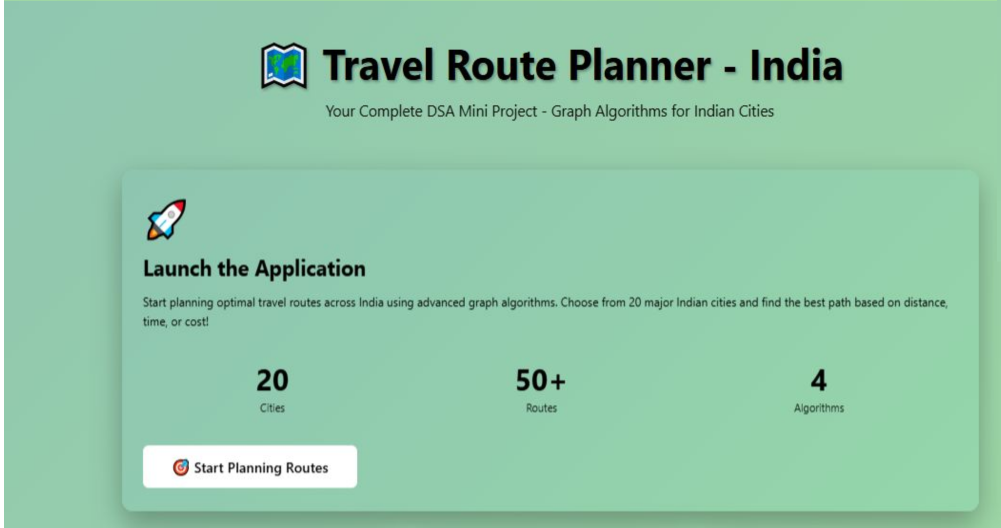
Real-World Applications

Navigation Systems
GPS, Google Maps

Logistics
Delivery optimization

Travel Industry
Flight planning

Network Routing
Internet packets



Travel Route Planner - India

Your Complete DSA Mini Project - Graph Algorithms for Indian Cities

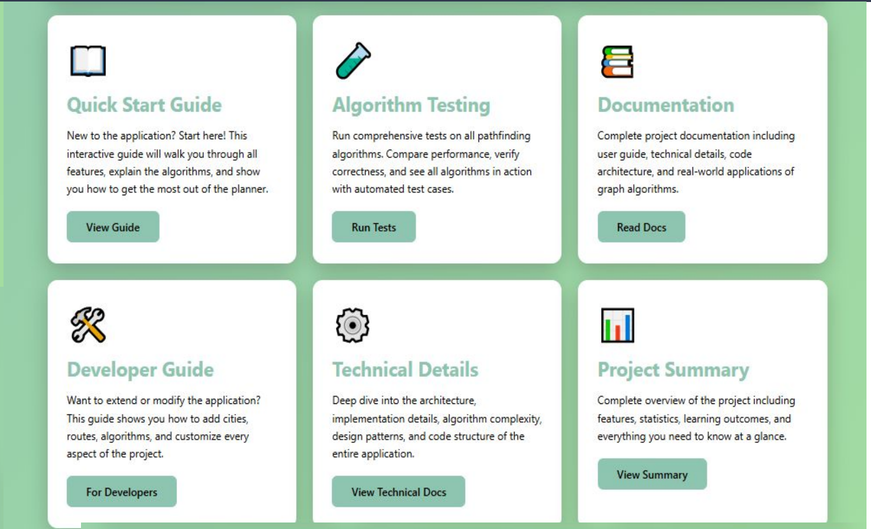
Launch the Application

Start planning optimal travel routes across India using advanced graph algorithms. Choose from 20 major Indian cities and find the best path based on distance, time, or cost!

20
Cities

50+
Routes

4
Algorithms



Quick Start Guide

New to the application? Start here! This interactive guide will walk you through all features, explain the algorithms, and show you how to get the most out of the planner.

Algorithm Testing

Run comprehensive tests on all pathfinding algorithms. Compare performance, verify correctness, and see all algorithms in action with automated test cases.

Documentation

Complete project documentation including user guide, technical details, code architecture, and real-world applications of graph algorithms.

Developer Guide

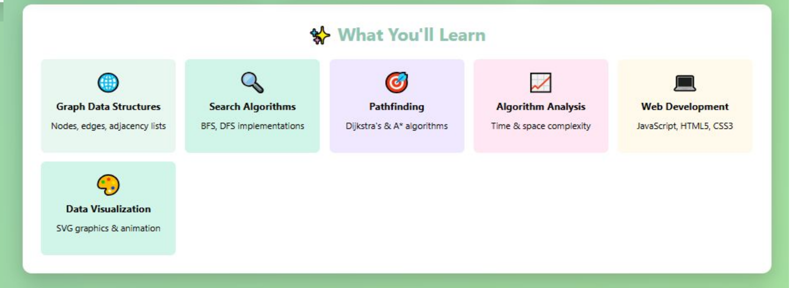
Want to extend or modify the application? This guide shows you how to add cities, routes, algorithms, and customize every aspect of the project.

Technical Details

Deep dive into the architecture, implementation details, algorithm complexity, design patterns, and code structure of the entire application.

Project Summary

Complete overview of the project including features, statistics, learning outcomes, and everything you need to know at a glance.



What You'll Learn

Graph Data Structures
Nodes, edges, adjacency lists

Search Algorithms
BFS, DFS implementations

Pathfinding
Dijkstra's & A* algorithms

Algorithm Analysis
Time & space complexity

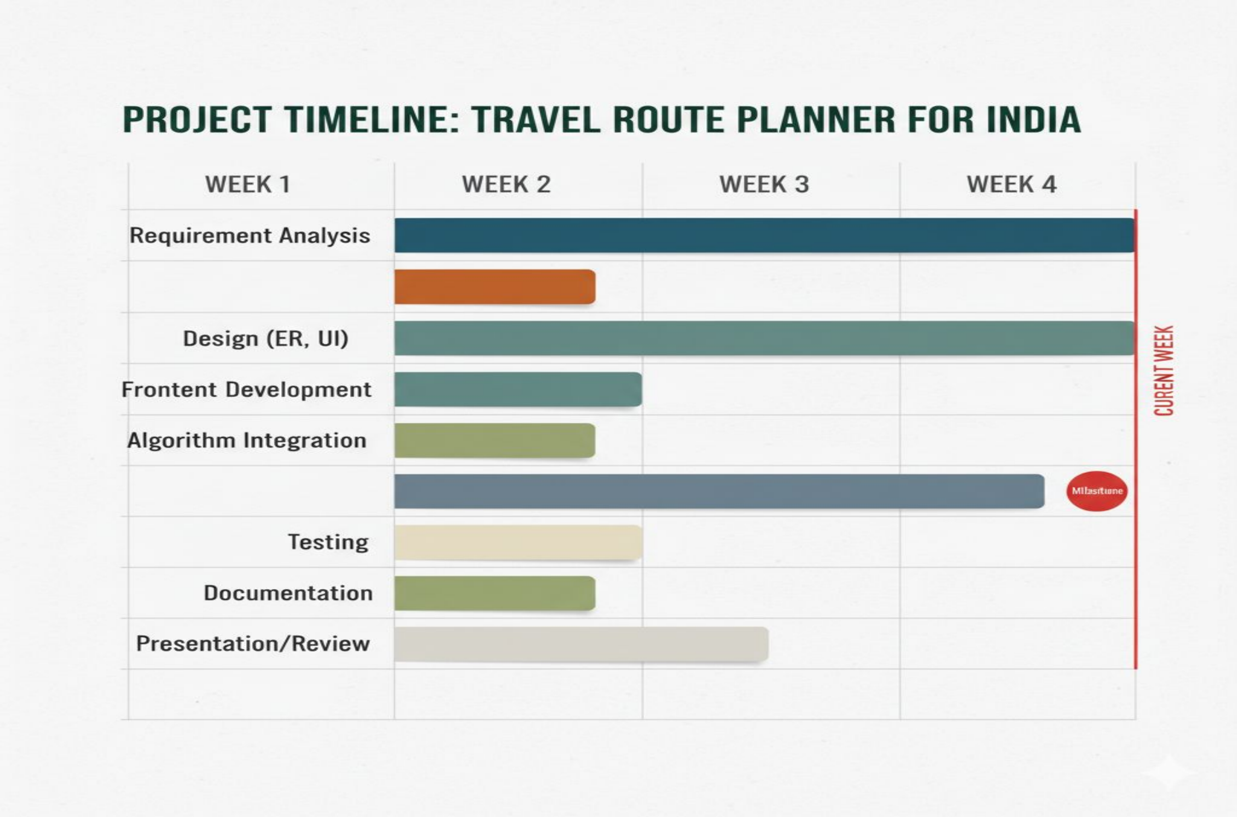
Web Development
JavaScript, HTML5, CSS3

Data Visualization
SVG graphics & animation

Implementation

- Open index.html in browser—no installation is required.
- Select the starting city, destination, and algorithm.
- See results and route visualization instantly.

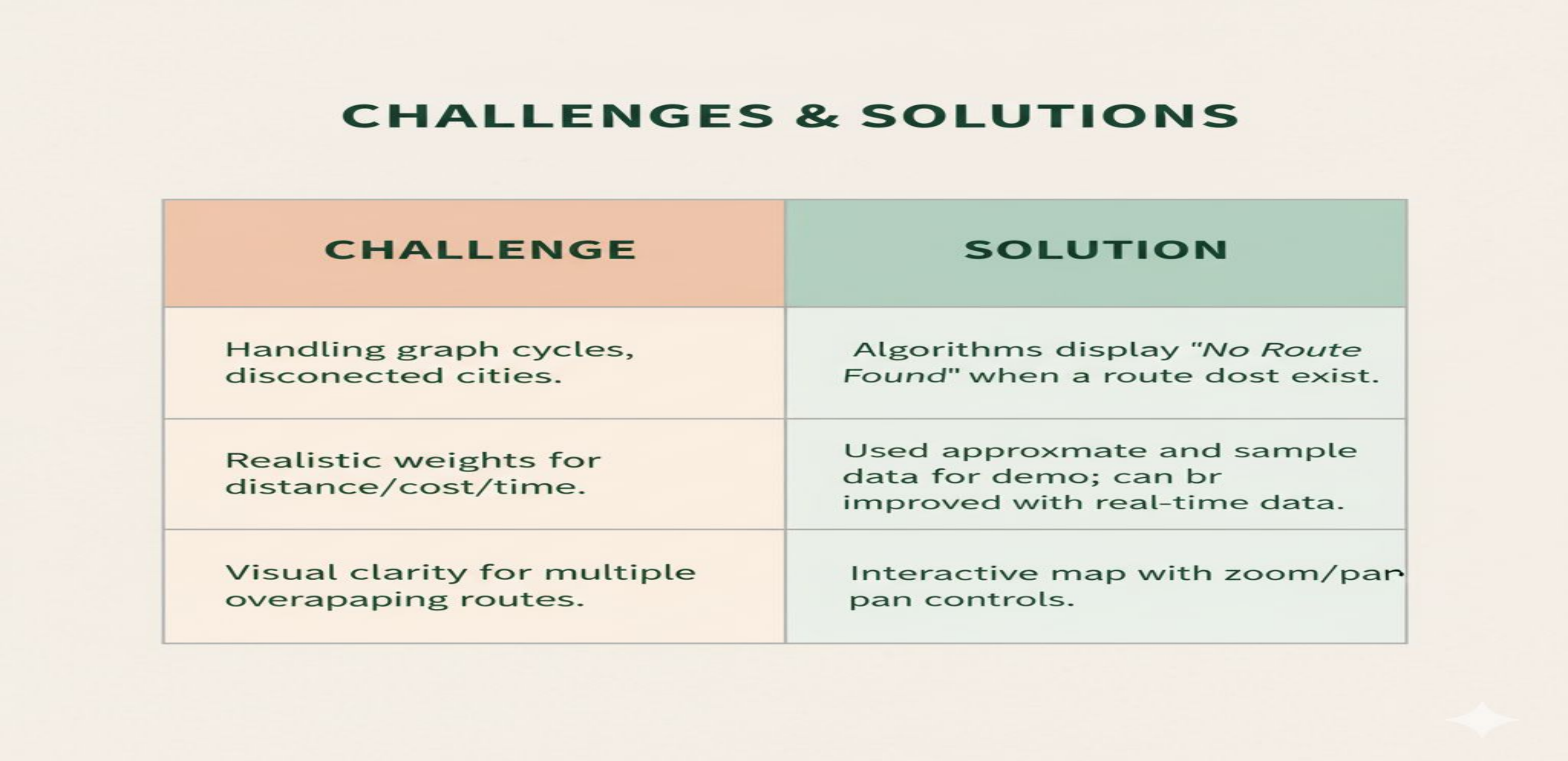
Gantt Chart


PROJECT TIMELINE: TRAVEL ROUTE PLANNER FOR INDIA
WEEK 1
WEEK 2
WEEK 3
WEEK 4
Requirement Analysis
Design (ER, UI)
Frontent Development
Algorithm Integration
Milasitione
Testing
Documentation
Presentation/Review
CURENT WEEK

Test Cases

- Select route from Mumbai to Kolkata with Dijkstra: Result matches expected optimal path.
- Try BFS and compare the number of route stops.
- Test all algorithms with different city pairs and criteria (distance, cost, time).
- Check invalid routes (disconnected cities): Appropriate error message shown.

Challenges & Solutions



CHALLENGES & SOLUTIONS

CHALLENGE	SOLUTION
Handling graph cycles, disconected cities.	Algorithms display *"No Route Found"* when a route dost exist.
Realistic weights for distance/cost/time.	Used approxmate and sample data for demo; can br improved with real-time data.
Visual clarity for multiple overapaping routes.	Interactive map with zoom/pan pan controls.

Future Scope

- Expand the project to include more cities/routes.
- Integrate with real-time APIs (Google Maps, Railways, Airlines) for live data.
- Add user accounts, personalized preferences, and multi-modal transport planning.

Code

Graph structure -

```
typedef struct {
City cities[MAX_CITIES];      // Array of 20 Indian cities
   AdjNode* adjList[MAX_CITIES];  // Adjacency list for routes
   int numCities, numRoutes; // Counters
} Graph;
```

Dijkstra’s algorithm -

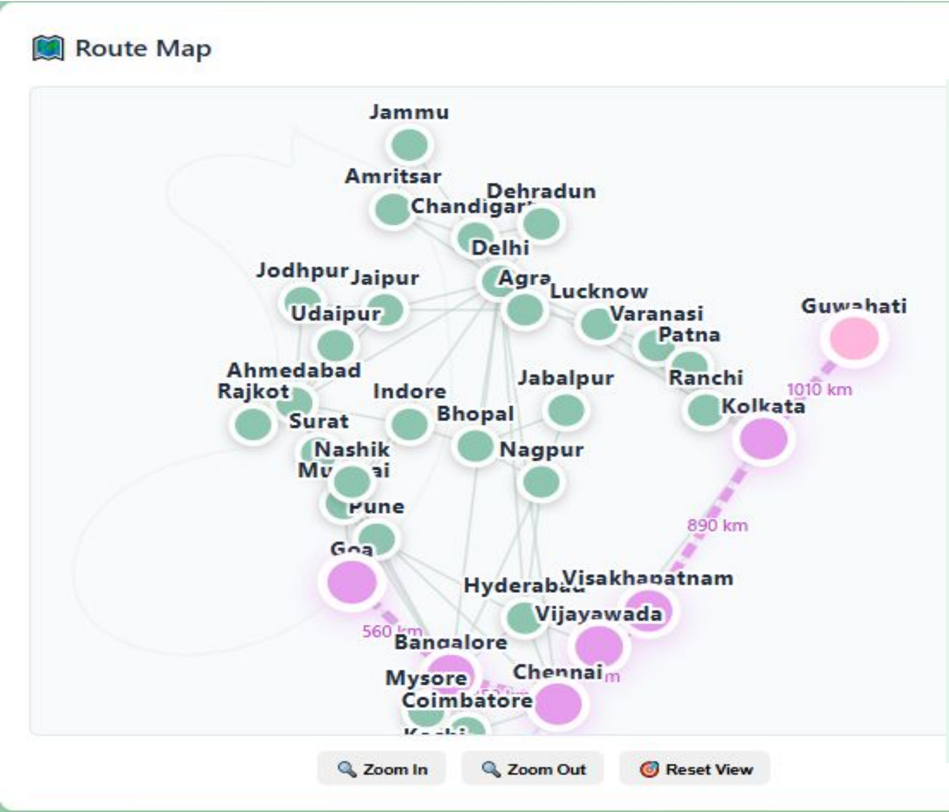
```
while (!isPQEmpty(pq)) {
   int current = dequeuePQ(&pq);
   // Explore neighbors and relax edges
   if (newDist < dist[next]) {
      dist[next] = newDist;
      parent[next] = current;
      enqueuePQ(&pq, next, newDist);
   }
}
```

BFS & DFS Comparison -

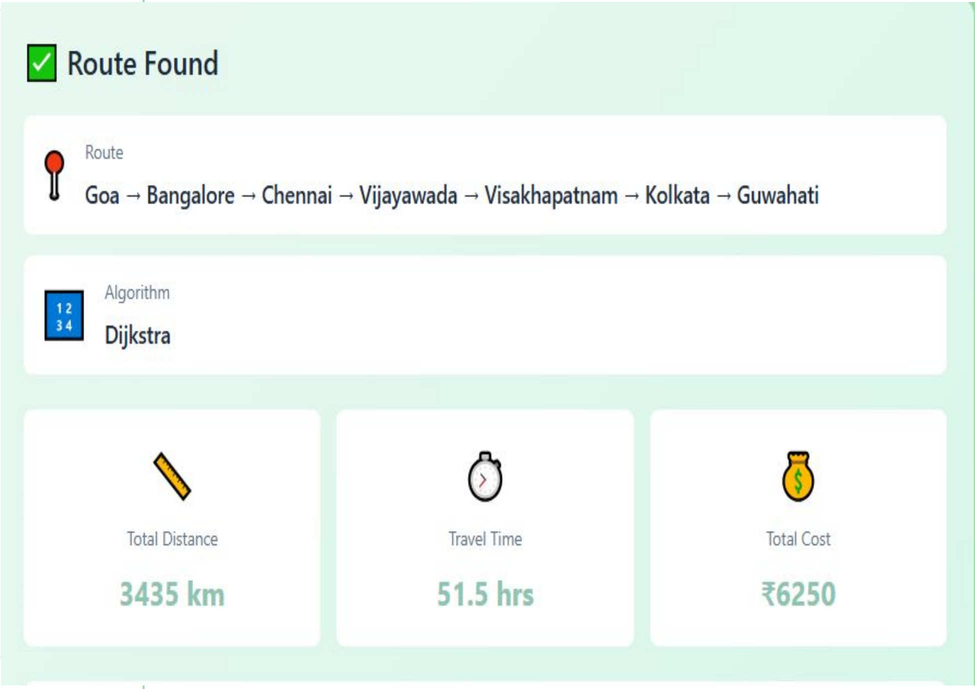
```
// BFS - Minimum hops (Queue-based)
queue[rear++] = start;
while (front < rear) {
   current = queue[front++];
   // Explore level-by-level
}

// DFS - Any path (Recursive)
bool dfsHelper(int current, int end) {
   if (current == end) return true;
   // Explore depth-first recursively
}
```

Output Screenshots



Route Map



✅ Route Found

Route

Goa → Bangalore → Chennai → Vijayawada → Visakhapatnam → Kolkata → Guwahati

Algorithm

Dijkstra

Total Distance

3435 km

Travel Time

51.5 hrs

Total Cost

₹6250

Conclusion

The project efficiently demonstrates how graph algorithms can solve real-world pathfinding problems in travel and logistics. Users gain insight into DSA implementation and visualization through a browser-based educational tool.

References

- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, “Introduction to Algorithms,” 3rd ed., MIT Press, 2009.
- J. Kleinberg and É. Tardos, “Algorithm Design,” Pearson, 2005.
- S. S. Skiena, “The Algorithm Design Manual,” 2nd ed., Springer, 2008.
- Mozilla Developer Network, “HTML5 Reference,” Available: https://developer.mozilla.org/
- “Dijkstra Algorithm.” GeeksforGeeks, Available: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-graph/