আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম
الجامعة الإسلامية العالمية شيتاغونغ
International Islamic University Chittagong

Department of Computer Science & Engineering

Project Report

Course Title: Computer Algorithms Lab

Course Code: CSE-2422

Project Name: Smart Delivery Route Optimizer

Submitted by:

Member 1 : Jumanah Rahman Anussha (C241472)

Member 2 : Jannatul Ferdous Tania (C241486)

Member 3: Tajia Jahan Tanha(C241502)

Section: 4CF

Submission Date: 20/01/2026

Submitted To:

Miskatul Jannat Tuly

Lecturer, Dept. of CSE, IIUC

Smart Delivery Route Optimizer

Introduction:

Delivery route optimization is the process of determining the most efficient sequence of stops for delivery personnel to minimize travel distance, time, and operational cost while satisfying real-world constraints such as vehicle capacity, service time, and geographic distribution. In practice, this transforms the classical Vehicle Routing Problem (VRP) — an NP-hard challenge in computer science — into a solvable, scalable solution using smart algorithmic strategies.

The Smart Delivery Route Optimizer is an intelligent system that goes beyond basic pathfinding. It dynamically assigns customer addresses to delivery personnel using K-Means Clustering, then selects the most appropriate routing algorithm based on cluster size: exact TSP with Dynamic Programming for tiny sets, Divide & Conquer for medium sets, and Greedy Nearest Neighbor for large sets. Furthermore, it incorporates real-world factors such as weather conditions, round-trip depot return, ASCII-based route visualization, performance competition, and a structured customer feedback mechanism — making it a complete, practical tool for modern logistics.

Objectives:

1. To develop a hybrid algorithm system that adaptively selects optimal routing strategies based on cluster size.

2. To incorporate real-world factors (weather, feedback, visualization) for practical logistics management.

Hardware and Software Requirements:

Hardware Requirements:
- Processor: Intel Core i5 or equivalent
- RAM: 8GB

Software Requirements:

- Operating System: Windows

- Compiler: C++

- IDE: Optional (Code::Blocks)

- Programming Language: C++
- Algorithms: K-Means, Greedy, TSP (DP), Divide & Conquer
- Concepts Used: Data Structures, OOP, Graph Algorithms

Algorithms Used and Their Roles in the System:

Below is a breakdown of each algorithm, its purpose, and where it is applied in the code:

Algorithm	When It Is Used	Why It Is Used
K-Means Clustering	After input collection, before routing	To group geographically close delivery addresses into k clusters (k = number of delivery boys), ensuring each boy serves a compact zone and reducing cross-area travel.
TSP with Dynamic Programming	Inside divideAndConquerRoute() when ≤ 3 addresses	Guarantees the shortest possible route for very small sets (optimal solution). It uses bitmask DP to explore all permutations efficiently.
Divide & Conquer Routing	When cluster size is ≤ 10	Recursively splits addresses by x-coordinate, solves subproblems, and merges routes. Provides near-optimal paths without exponential cost.
Greedy Nearest Neighbor	When cluster size is > 10	At each step, visits the nearest unvisited address. Fast and practical for large sets where optimal solutions are computationally infeasible.
Weather Adjustment Logic	After base route optimization	Applies speed/time multipliers (e.g., 0.7× speed in rain) to reflect real-world delays and improve time estimation accuracy.
Feedback Analysis Engine	In the post-delivery interactive menu	Collects customer ratings, auto-tags issues, and generates performance reports for continuous service improvement.

Algorithm Mapping And Complete System Workflow:

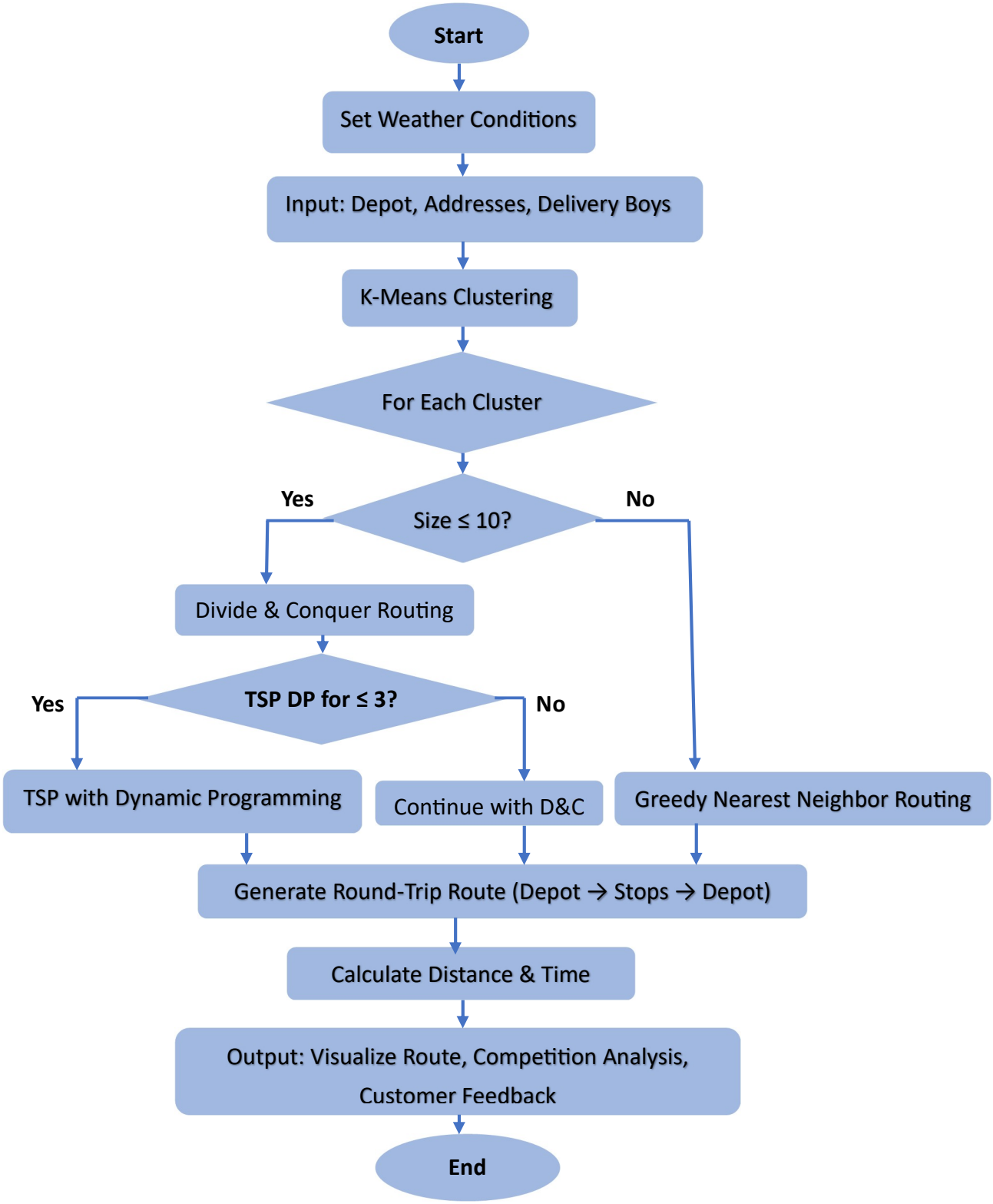
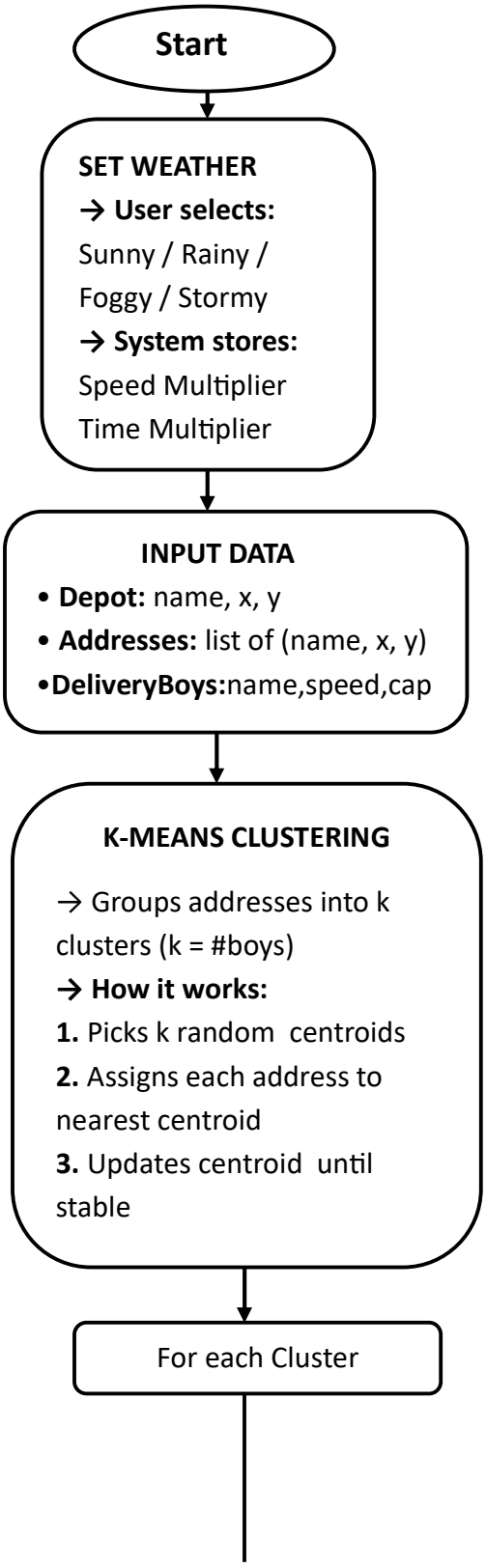
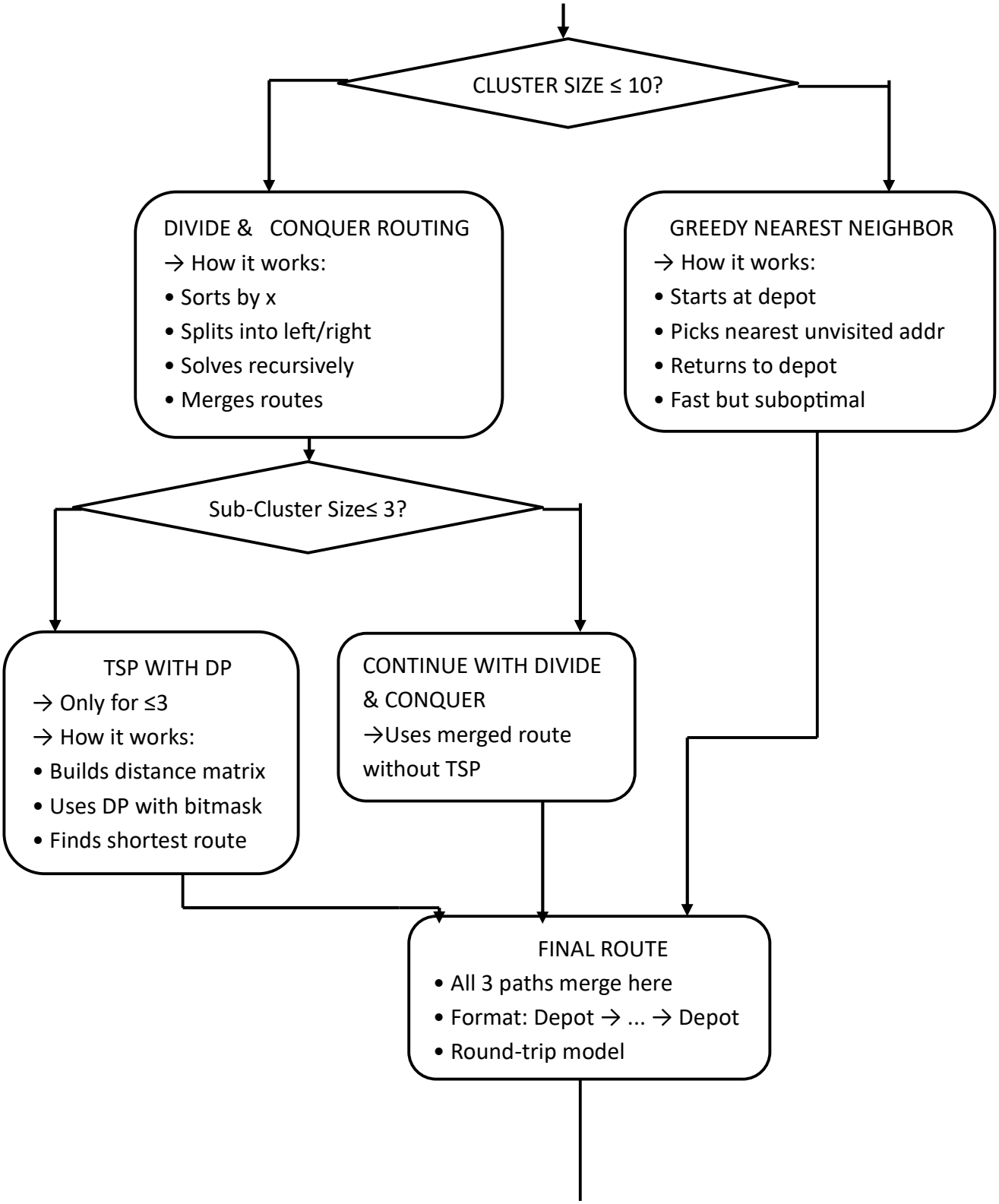


Fig-01: Algorithm Selection Workflow


Start
SET WEATHER
→ User selects:
Sunny / Rainy /
Foggy / Stormy
→ System stores:
Speed Multiplier
Time Multiplier
INPUT DATA
• Depot: name, x, y
• Addresses: list of (name, x, y)
•DeliveryBoys:name,speed,cap
K-MEANS CLUSTERING
→ Groups addresses into k clusters (k = #boys)
→ How it works:
1. Picks k random centroids
2. Assigns each address to nearest centroid
3. Updates centroid until stable
For each Cluster


CLUSTER SIZE ≤ 10?
DIVIDE & CONQUER ROUTING
→ How it works:
• Sorts by x
• Splits into left/right
• Solves recursively
• Merges routes
GREEDY NEAREST NEIGHBOR
→ How it works:
• Starts at depot
• Picks nearest unvisited addr
• Returns to depot
• Fast but suboptimal
Sub-Cluster Size≤ 3?
TSP WITH DP
→ Only for ≤3
→ How it works:
• Builds distance matrix
• Uses DP with bitmask
• Finds shortest route
CONTINUE WITH DIVIDE & CONQUER
→Uses merged route without TSP
FINAL ROUTE
• All 3 paths merge here
• Format: Depot → ... → Depot
• Round-trip model

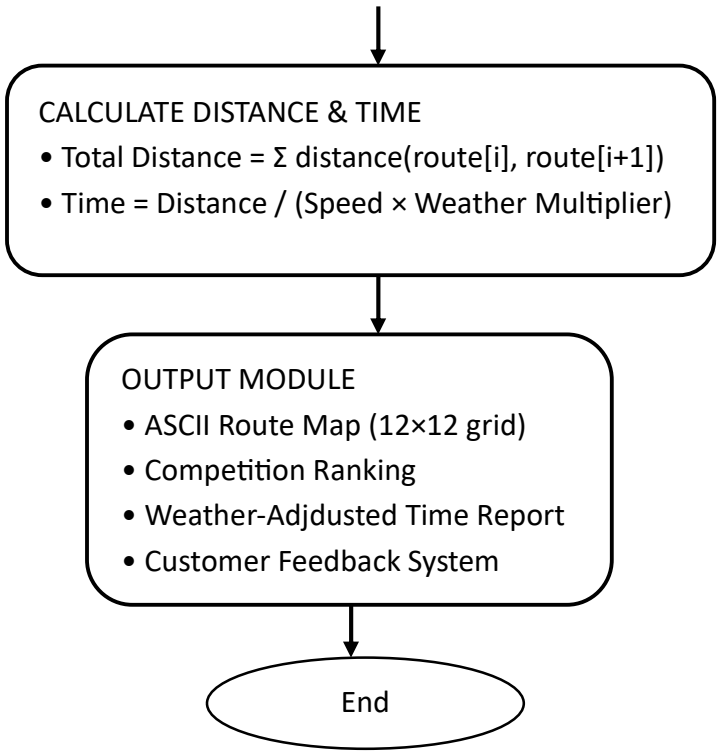


Fig-02 : Smart Delivery Route Optimizer - Main System Workflow

Algorithm Selection Decision Tree:

Addresses → K-Means → Clusters

↓

Check Cluster Size

/ | \

≤5 ≤10 >10

| | |

TSP Divide & Conquer Greedy (Nearest Neighbor)

Fig -04: Decision Tree

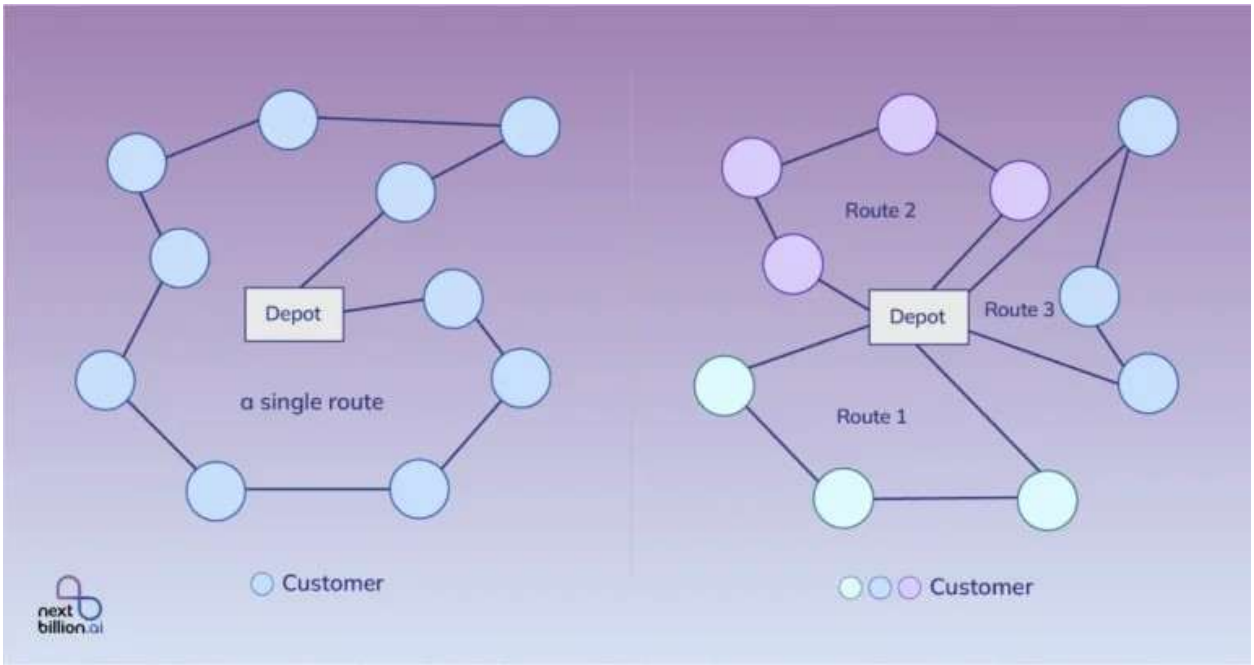


Fig -04: Delivery Execution & Analysis Process

Code:

https://github.com/taniajannat12/Projects/tree/main/Computer%20Algorithms

https://github.com/tajia-aslanbey/C.Algorithm-Project.git

https://github.com/anussha2023/Computer-Algorithms/tree/main/Algorithm_CSE-2422

Implementation Details:

This C++ program implements a **Delivery Route Optimization System**.

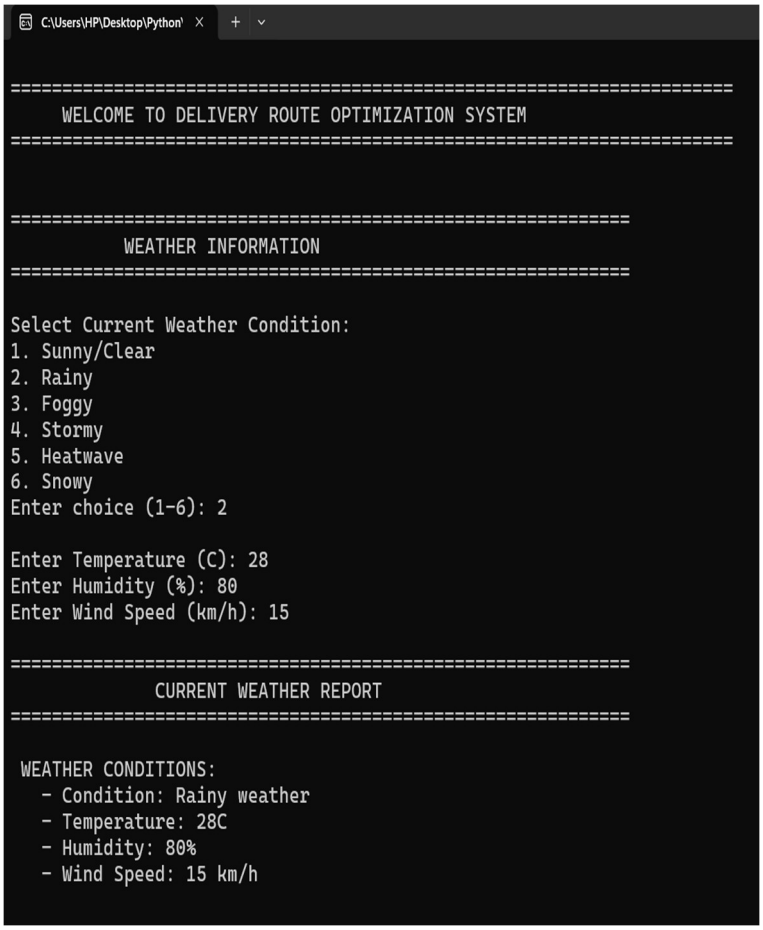
The code:

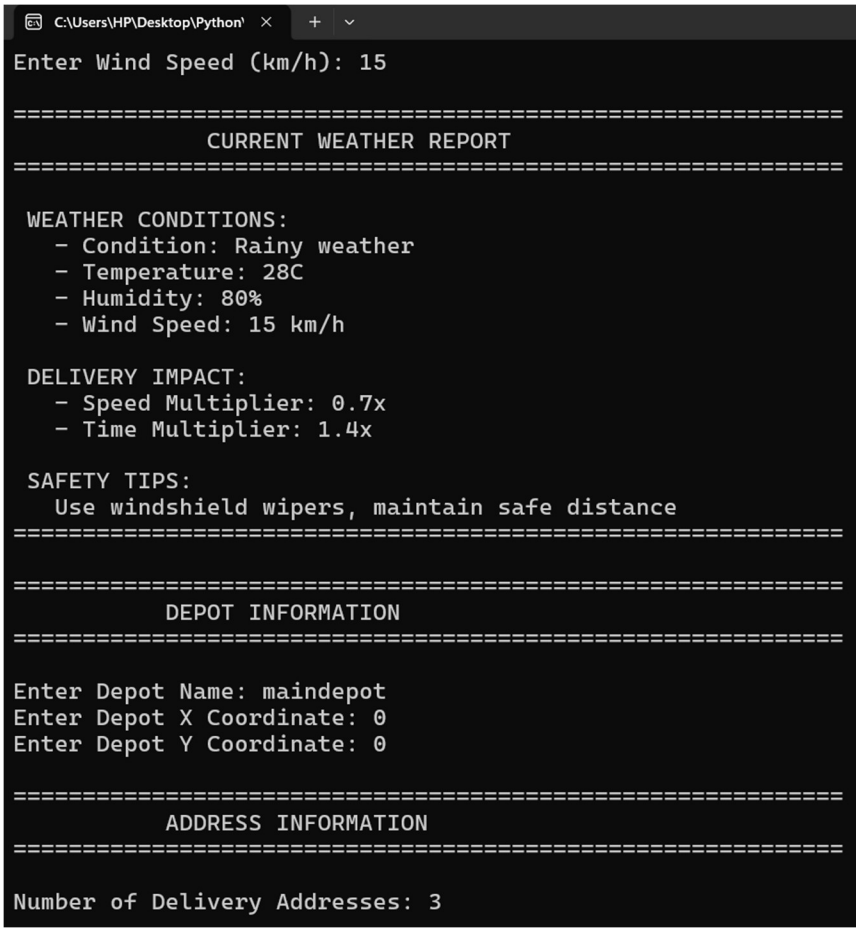
- Defines structures for **Address**, **Depot**, and **DeliveryBoy**
- Calculates distance between locations using **Euclidean distance**
- Uses **K-Means Clustering** to divide delivery addresses among delivery boys
- Applies **Greedy algorithm** and **Divide & Conquer (TSP-based)** methods to find optimized delivery routes
- Ensures each route **starts and ends at the depot**
- Visualizes routes using a simple grid-based map
- Compares delivery boys based on **total distance and delivery time**

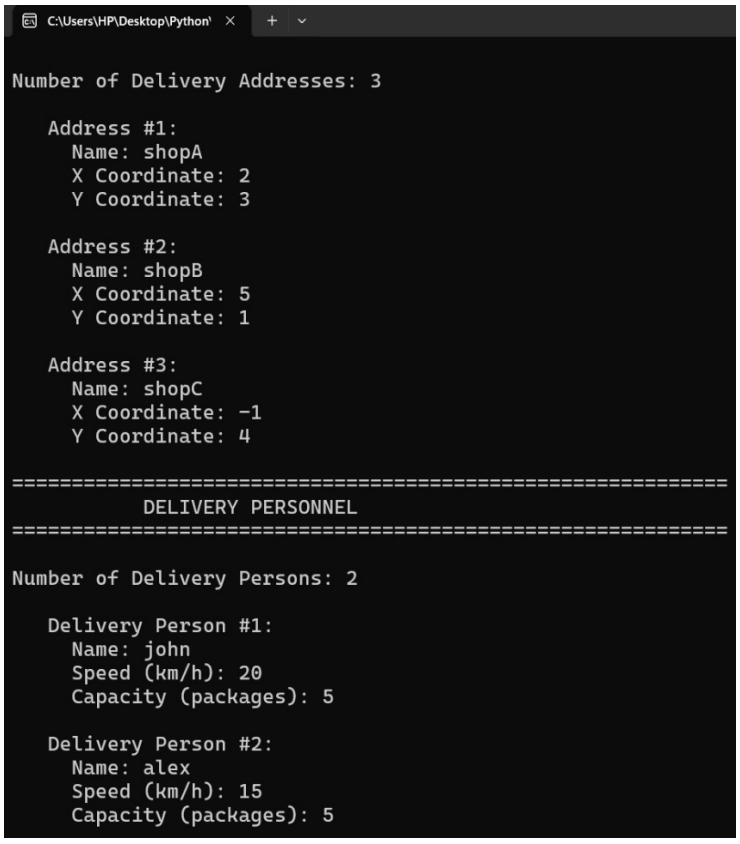
- Adjusts delivery time according to **weather conditions**
- Collects **customer feedback**, calculates ratings, and generates performance reports

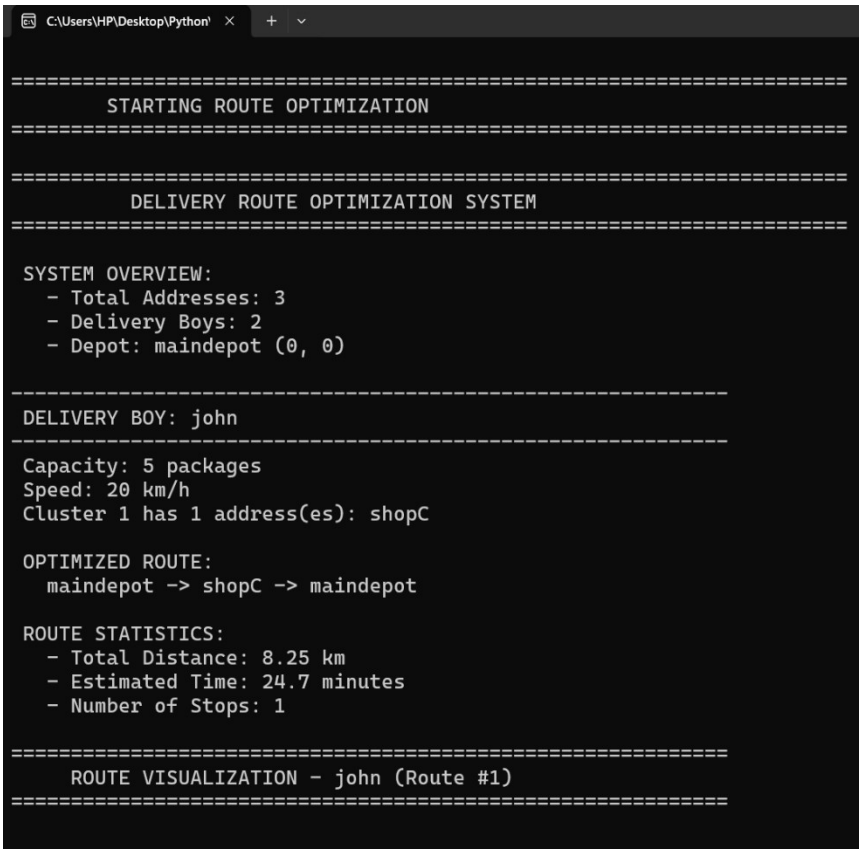
The main() function takes all inputs, runs route optimization, applies weather effects, and manages the feedback system.

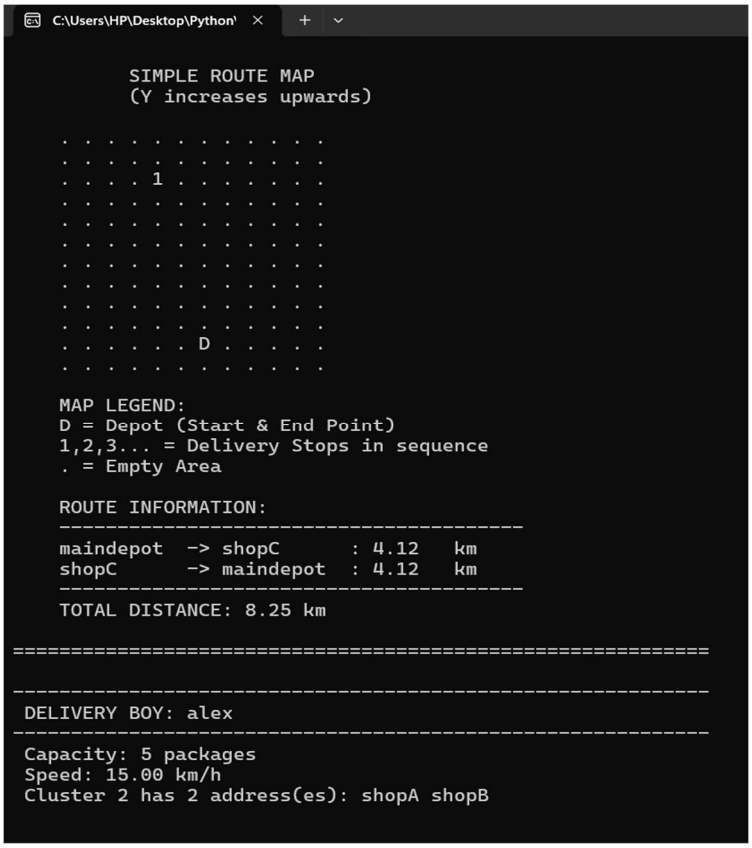
Output:

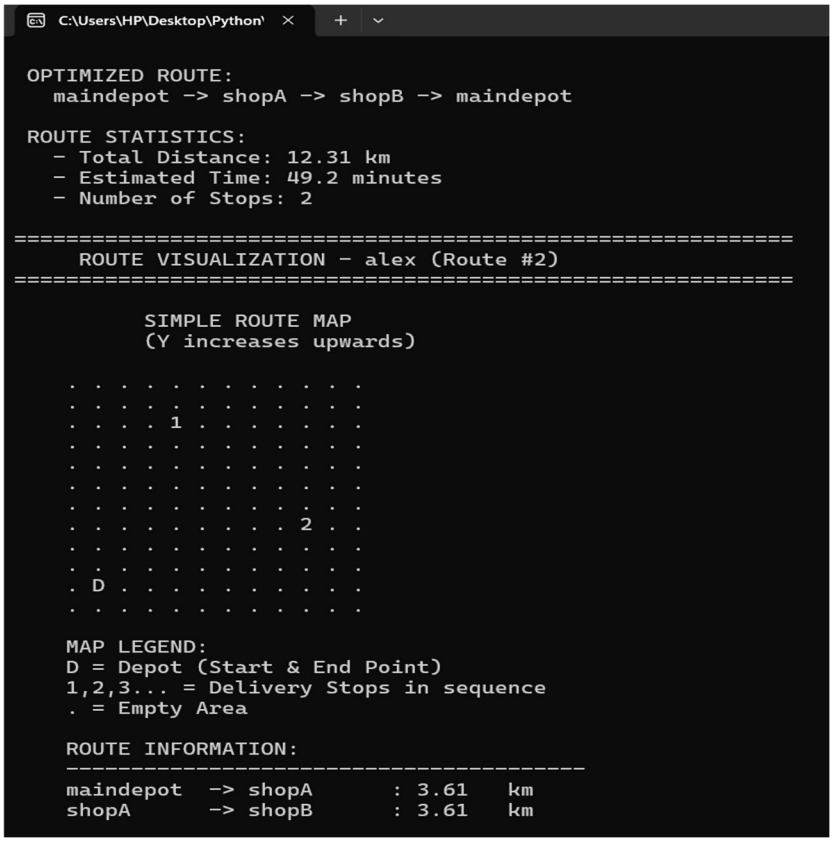


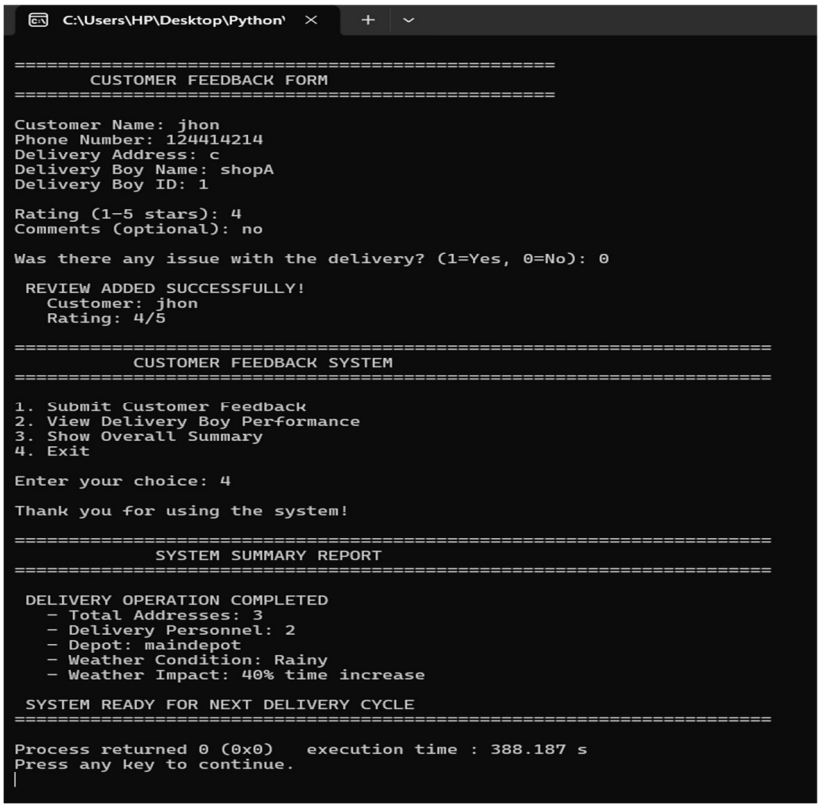



C:\Users\HP\Desktop\Python\
Number of Delivery Addresses: 3
Address #1:
Name: shopA
X Coordinate: 2
Y Coordinate: 3
Address #2:
Name: shopB
X Coordinate: 5
Y Coordinate: 1
Address #3:
Name: shopC
X Coordinate: -1
Y Coordinate: 4
DELIVERY PERSONNEL
Number of Delivery Persons: 2
Delivery Person #1:
Name: john
Speed (km/h): 20
Capacity (packages): 5
Delivery Person #2:
Name: alex
Speed (km/h): 15
Capacity (packages): 5


C:\Users\HP\Desktop\Python\
STARTING ROUTE OPTIMIZATION
DELIVERY ROUTE OPTIMIZATION SYSTEM
SYSTEM OVERVIEW:
- Total Addresses: 3
- Delivery Boys: 2
- Depot: maindepot (0, 0)
DELIVERY BOY: john
Capacity: 5 packages
Speed: 20 km/h
Cluster 1 has 1 address(es): shopC
OPTIMIZED ROUTE:
maindepot -> shopC -> maindepot
ROUTE STATISTICS:
- Total Distance: 8.25 km
- Estimated Time: 24.7 minutes
- Number of Stops: 1
ROUTE VISUALIZATION - john (Route #1)


C:\Users\HP\Desktop\Python\
SIMPLE ROUTE MAP
(Y increases upwards)
1
D
MAP LEGEND:
D = Depot (Start & End Point)
1,2,3... = Delivery Stops in sequence
. = Empty Area
ROUTE INFORMATION:
maindepot -> shopC : 4.12 km
shopC -> maindepot : 4.12 km
TOTAL DISTANCE: 8.25 km
DELIVERY BOY: alex
Capacity: 5 packages
Speed: 15.00 km/h
Cluster 2 has 2 address(es): shopA shopB


C:\Users\HP\Desktop\Python\
OPTIMIZED ROUTE:
maindepot -> shopA -> shopB -> maindepot
ROUTE STATISTICS:
- Total Distance: 12.31 km
- Estimated Time: 49.2 minutes
- Number of Stops: 2
ROUTE VISUALIZATION - alex (Route #2)
SIMPLE ROUTE MAP
(Y increases upwards)
1
2
D
MAP LEGEND:
D = Depot (Start & End Point)
1,2,3... = Delivery Stops in sequence
. = Empty Area
ROUTE INFORMATION:
maindepot -> shopA : 3.61 km
shopA -> shopB : 3.61 km



Output Analysis:

Distance Calculations:

Distances are calculated using Euclidean formula:

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

John's Route:

- Depot → shopC = 4.12 km
- shopC → Depot = 4.12 km
- Total Distance = 8.25 km

Alex's Route:

- Depot → shopA = 3.61 km
- shopA → shopB = 3.61 km
- shopB → Depot = 5.10 km
- Total Distance = 12.31 km

2️⃣ Delivery Time Calculation

Base time = Total distance ÷ Speed × 60 (minutes)

John: $24.7 \text{ min} = 8.25/20 \times 60$

Alex: $49.2 \text{ min} = 12.31/15 \times 60$

3️⃣ Weather-Adjusted Time

Weather multiplier (Rainy) = 1.4 × base time

John: $24.7 \times 1.4 \approx 42.0$ min

Alex: $49.2 \times 1.4 \approx 56.0$ min

Impact: 40% increase in delivery time due to rain.

4️⃣ Time Gap Between Delivery Boys
Time Gap $= 49.2 - 24.7 = 24.5$ min

5️⃣ Customer Feedback Example

- Customer gave 4/5 rating for "shopA" delivery.
- No issues reported.

Delivery Operation Summary

- Total Addresses: 3
- Delivery Personnel: 2 (John, Alex)
- Depot: maindepot
- Total Distances: John = 8.25 km, Alex = 12.31 km
- Base Delivery Times: John = 24.7 min, Alex = 49.2 min
- Weather-Adjusted Times: John = 42.0 min, Alex = 56.0 min (+40%)
- Time Gap: 24.5 min (Alex behind John)
- Winner: John – returned to depot first
- Customer Feedback: 1 review, rating 4/5, no issues reported

Complexity Analysis:

The system uses multiple algorithmic techniques such as K-Means clustering, Greedy heuristics, Divide & Conquer, and Dynamic Programming. Each algorithm is applied based on input size to maintain efficiency and scalability.

Distance Calculation: double distance(const Address &a, const Address &b)

(The distance between two points is computed using a fixed number of arithmetic operations.)

Time Complexity: O(1)

Space Complexity: O(1)

K-Means Clustering: kMeansClustering(addresses, k)

(Assign each address to nearest centroid → O(nk)

Update centroids → O(n)

Repeated for max I = 100 iterations (constant))

Time Complexity: $(I \times n \times k) \approx O(nk)$
Space Complexity: O(n+k)

Greedy Route Optimization (Nearest Neighbor): greedyRoute(addresses, start)

(For each address, find nearest unvisited address)

Time Complexity: O(n^2)
Space Complexity: O(n)

TSP using Dynamic Programming (Bitmask DP): tspDP(distMatrix, start)

(The algorithm evaluates all possible subsets of nodes using bitmask-based dynamic programming.)

Time Complexity: O(n^2 × 2^n)
Space Complexity: O(n×2^n)

Divide & Conquer Route Optimization: divideAndConquerRoute(addresses, start)

(The address list is recursively divided into smaller parts and merged after sorting.
Recurrence Relation: T(n)=2T(n/2)+O(n))
Time Complexity: O(nlogn)
Space Complexity: O(n)

Route Visualization: visualizeRoute(route)

(Grid mapping + distance calculations)

Time Complexity: O(n)

Space Complexity: O(1)

Competition Analysis : analyzeCompetition(routes, boys)

(Distance calculation per route)

Time Complexity: O(n)

Space Complexity: O(k)

Weather Adjustment System : WeatherManager

(Weather impact calculations involve only constant-time operations.)

Time Complexity: O(1)

Space Complexity: O(1)

Customer Feedback System :

(Feedback data is stored and sorted to generate performance summaries.)

Time Complexity: O(r log r)

Space Complexity: O(r)

Complexity Table:

Module	Time Complexity	Space Complexity
Distance Calculation	O(1)	O(1)
K-Means Clustering	O(nk)	O(n + k)
Greedy Route Optimization	$O(n^2)$	O(n)
TSP (Dynamic Programming)	$O(n^2 \cdot 2^n)$	$O(n \cdot 2^n)$
Divide & Conquer Routing	O(n log n)	O(n)
Visualization	O(n)	O(1)
Weather System	O(1)	O(1)
Feedback System	O(r log r)	O(r)

TIME COMPLEXITY:

BEST: O(n log n)

WORST: $O(n^2)$

AVERAGE: O(n log n)

SPACE COMPLEXITY:

BEST: O(n) (Small clusters, no TSP)

WORST: $O(n \cdot 2^n)$ (TSP for $n \leq 5$, practical limit)

AVERAGE: O(n + k) (typical case)

Overall Time Complexity:

$O(n^2)$

Moderate time complexity. Works well for typical delivery scenarios (≤1000 addresses) but may need optimization for city-scale operations.

Overall Space Complexity:

O(n)

"Efficient memory usage. Linear growth ensures the system can handle increasing addresses without memory issues."

"A practical delivery system with efficient memory usage and acceptable time complexity for real-world medium-scale operations. Could benefit from algorithmic improvements for large scale deployment."

Challenges Faced And Future Scope:

Advantages

- Optimizes delivery routes efficiently
- Reduces total distance and delivery time
- Manages multiple delivery persons easily
- Adjusts delivery time based on weather conditions
- Evaluates performance using customer feedback

Limitations:

- Does not use real road maps
- Traffic conditions are not considered
- Performance may decrease for large datasets
- No real-time or internet-based data support
- **Delivery output times are sometimes inaccurate**
- **Route visualization does not always display correctly**; proper map visualization will be implemented in future

Future Improvements:

- Integration with Google Maps or real-world maps

- Include real-time traffic information
- Develop a mobile application
- Use AI-based advanced route optimization
- Add map integration for real navigation and live traffic updates for accurate delivery time predictions
- Fix output time calculations and improve route visualization
- **Console map is rough, may overlap points, not precise for exact distances.**

Conclusion:

The Delivery Route Optimization System efficiently manages multiple delivery personnel, optimizes routes to reduce total distance and time, and adjusts delivery estimates based on weather conditions. While the system currently lacks real road maps, traffic data, and accurate route visualization, it provides a strong foundation for future improvements such as integration with real-world maps, AI-based route optimization, and real-time traffic updates. Overall, this project demonstrates the potential of combining route optimization algorithms, clustering, and customer feedback to improve delivery efficiency and performance.

References:

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press. – K-Means, TSP, greedy algorithms, algorithm complexity.
- Kleinberg, J., & Tardos, É. (2006). *Algorithm Design*. Pearson. – Divide-and-conquer and clustering strategies.
- MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. – Original K-Means algorithm.
- Lawler, E. L., et al. (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley. – TSP dynamic programming.
- GeeksforGeeks: K-Means Clustering in C++ – https://www.geeksforgeeks.org/k-means-clustering/
- GeeksforGeeks: Traveling Salesman Problem using DP – https://www.geeksforgeeks.org/dsa/travelling-salesman-problem-using-dynamic-programming/