Logistics Management System

Name: P.D.U.Kavisha

Index Number: AS20240494

Course Code: CSC 1012 – Introduction to Computer Programming

GitHub Repository URL: https://github.com/UshaniKavisha/logistics-management-system.git

Submission Date: 26.10.2025

Contents

1. Introduction

This project is a menu-driven logistics management system developed using the C programming language for the course CSC 1012. In this program key programming concepts are applied key programming concepts such as arrays, functions, loops, conditionals, to simulate a simple logistics and delivery management system that handles cities, delivery routes, vehicle types, fuel consumption, and order costs.

2. Objective

The objective of this project is to design and implement a logistics management system that can:

- Manage cities and distances between them
- Handle customer delivery requests
- Estimate delivery time and cost
- Track completed deliveries
- Generate performance reports

3. System Overview

The Logistics Management System allows users to add, rename, remove cities, enter distances between cities, and select vehicles for deliveries. It can also find the least-cost (least-distance) route between cities.

In addition, the system generates a performance report that includes the total number of deliveries completed, total distance covered, average delivery time, total revenue and total profit, and the longest and shortest routes completed.

4. System Functional Requirements

- City Management
    - Store up to 30 cities.
    - Each city must have a unique name.
    - Functions: addCity(), renameCity(), removeCity(), listCities.

- Distance Management
    - Use a 2D array to store distances between cities (int distance [MAX_CITIES] [MAX_CITIES]).
    - Ensure distance are symmetrical (distance[i] [j] = distance[j] [i]).
    - Prevent invalid entries (distance to itself = 0).
    - Functions: initializeDistance (), enterDistance(), displayDistance().
    - Display the distance table neatly formatted.

- Vehicle Management
    - Three vehicle types: Van, Truck, and Lorry.
    - Store their capacity (kg), Rate per km (LKR), Average Speed (km/h) and Fuel Efficiency (km/l).
    - Functions: displayVehicle(), chooseVehicle().

- Delivery Request Handling
    - Input source city index, destination city index, weight, and vehicle type (0=Van, 1=Truck, 2=Lorry).
    - Validate that the weight does not exceed vehicle capacity and source ≠ destination.
    - Function: deliveryRequestHandling().

- Cost, Time, and Fuel Calculations
    - Delivery Cost = Distance × Rate × (1 + Weight/10000)
    - Estimated Delivery Time (hours) = Distance / Speed

- Fuel Consumption = Distance / Efficiency
- Fuel Cost = Fuel Consumption × Fuel price
- Total Operational Cost = Delivery Cost + Fuel Cost
- Profit = 25% markup on base cost
- Customer Charge = Total Cost + Profit

- Finding The Least-Cost Route (Least-Distance)

  - Finds the shortest route between two cities.
  - Compares all possible paths for up to 4 cities.
  - Function: findLeastCostRoute(startIndex, endIndex).

- Performance Reports

  - Total deliveries, total distance, average time, total profit and the longest and shortest routes distance.
  - Function: performanceReport().

- File Handling
  - Maintain .txt files to store data.
  - Functions: loadRoutes(),saveRoutes(),loadDeliveries(),saveDeliveries().

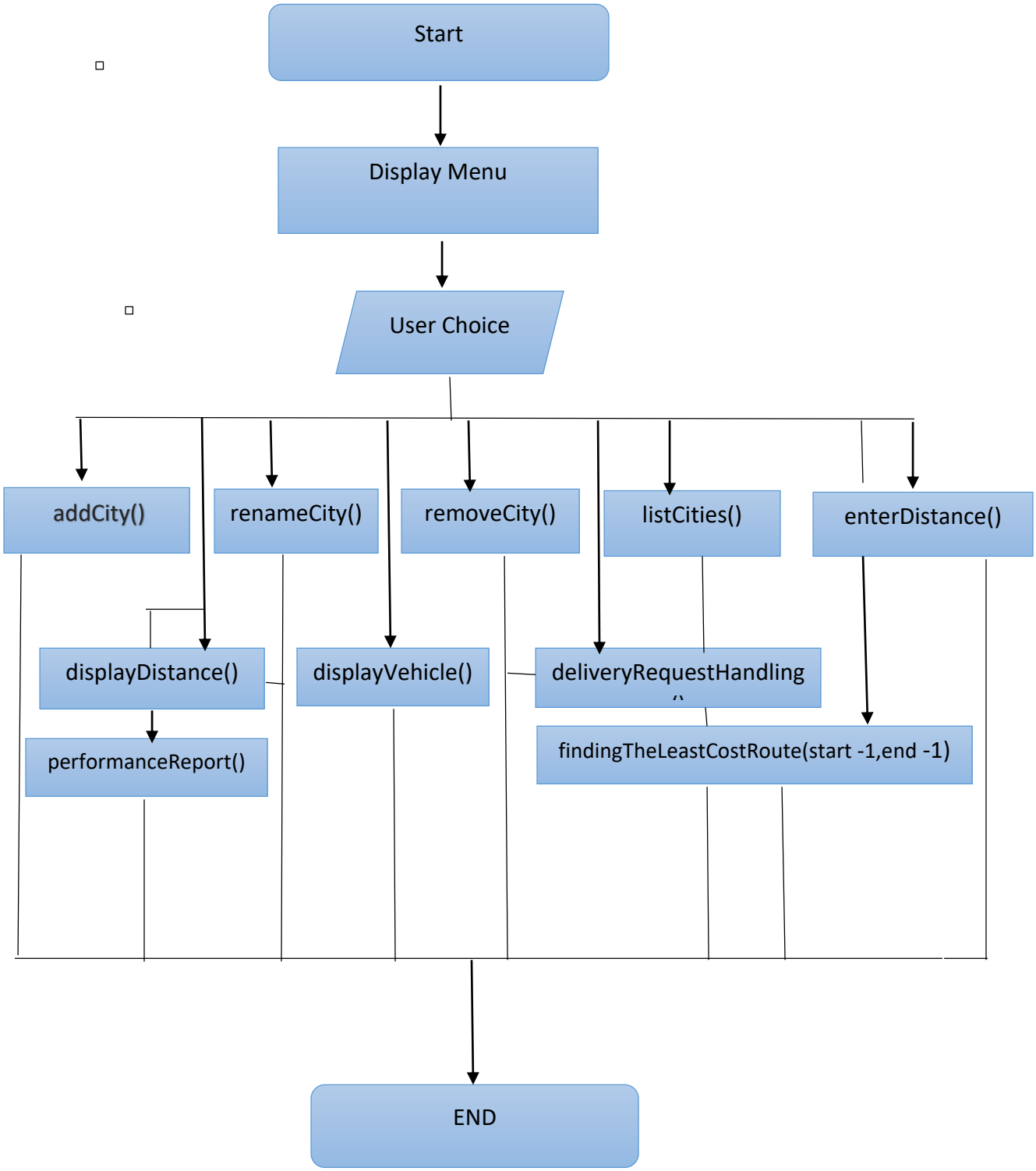
5. Program Structure

The program is modular and divided into several functions. Each function performs a specific task for better readability and maintenance.
Example function structure:

```
void addCity();
void initializeDistance();
void deliveryRequestHandling();
void findLeastCostRoute();
void performanceReport();
```

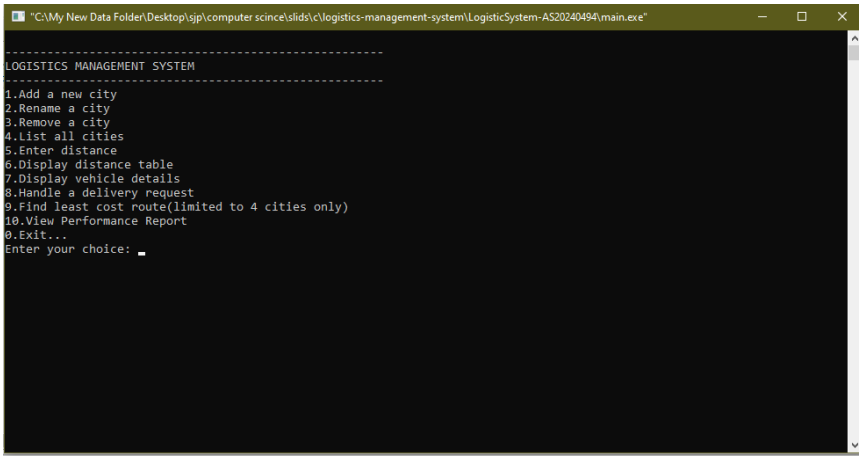
The main() function displays a menu and calls the relevant functions based on user selections.

6. Program Flowchart



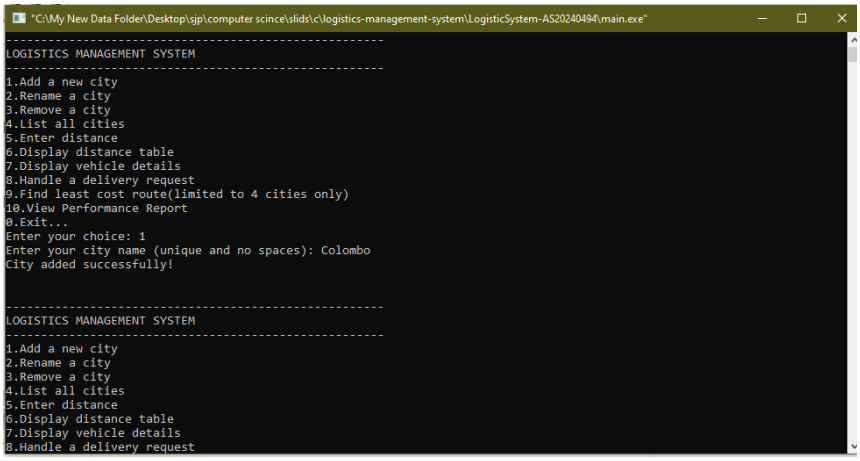
7. Sample Outputs and Screenshots

Figure 1: Main Menu



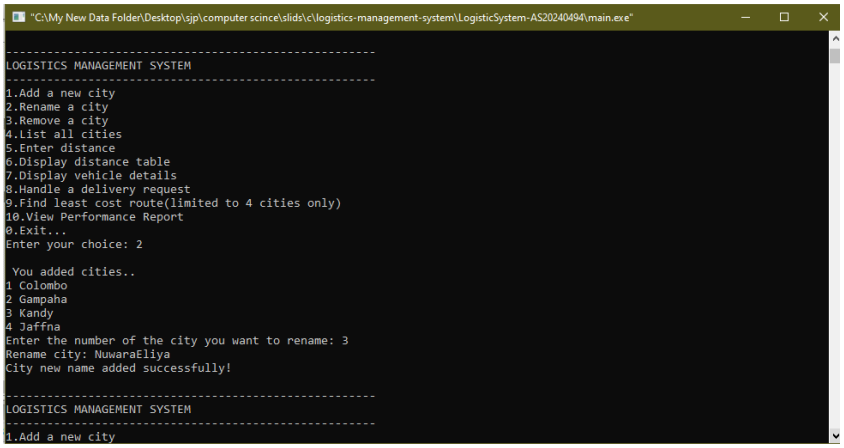
Displays all available options such as city management, delivery handling, and reports.

Figure 2: Adding Cities



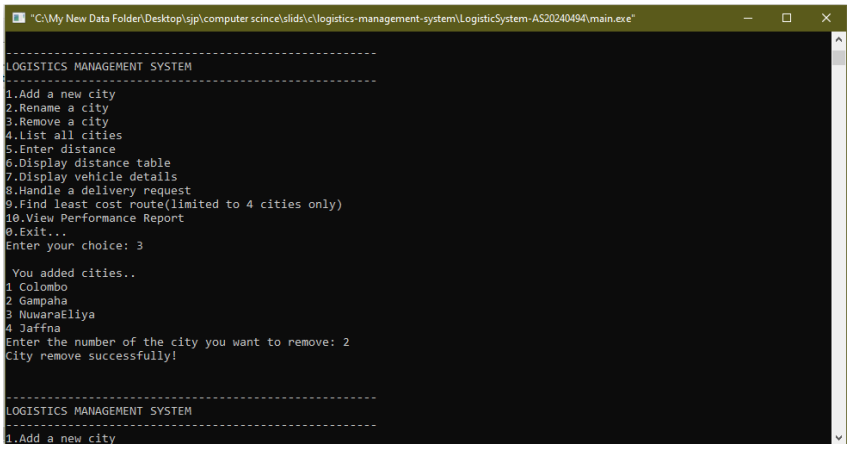
Shows how new cities are added to the system.

Figure 3: Rename Cities



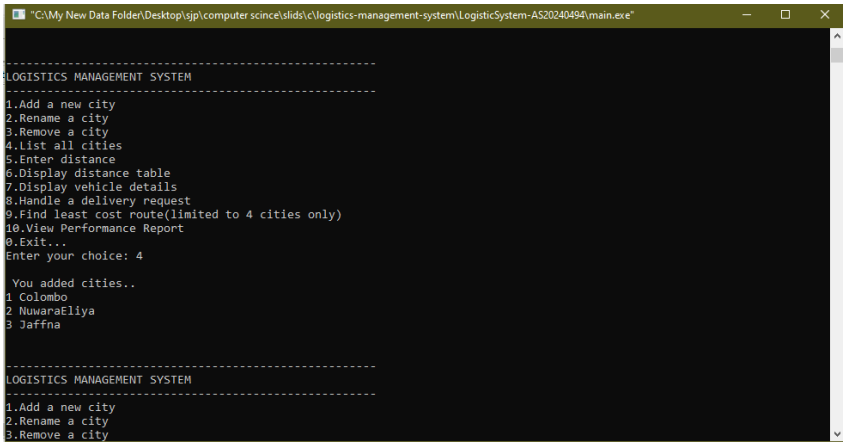
Shows how the program renames a city using the Rename City option.

Figure 4: Remove Cities



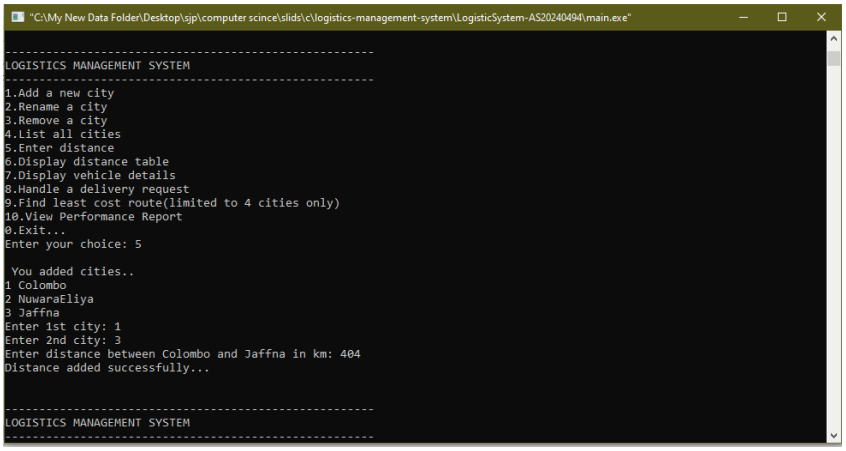
Shows how the program Remove City option.

Figure 5: List All Cities



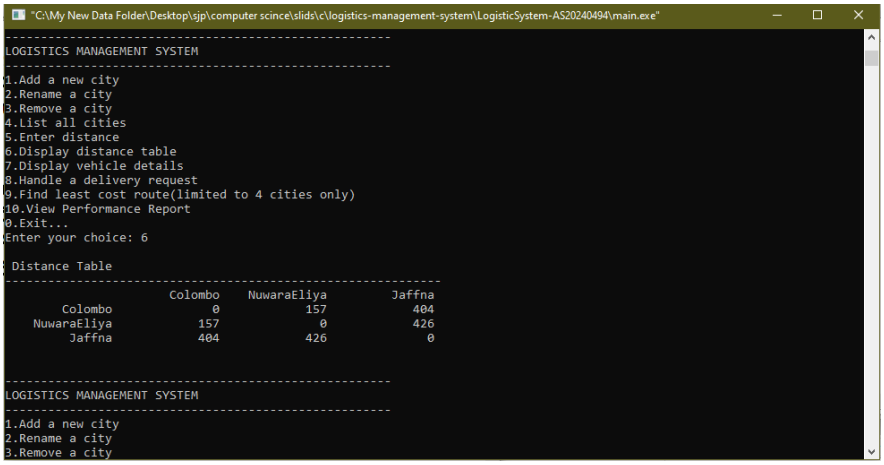
Displays all added cities currently stored in the system

Figure 6: Enter Distance



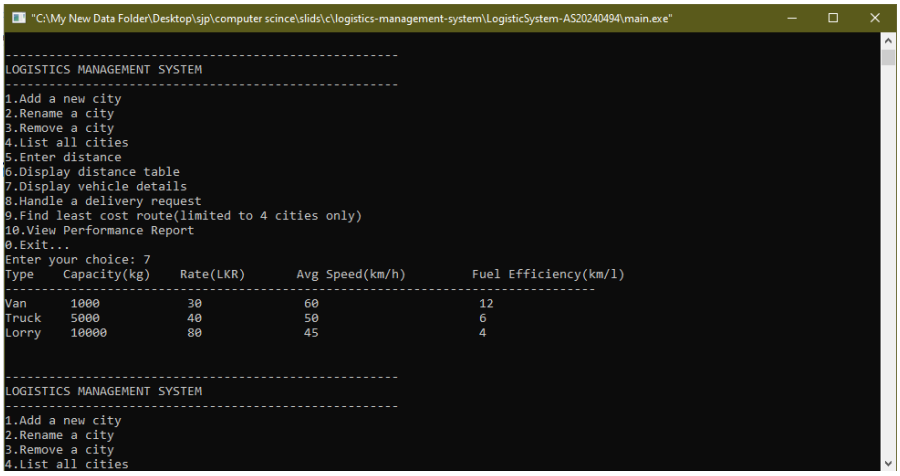
Shows how the user enters or edits the distance between two selected cities.

Figure 7: Distance Table Display



Displays the full distance table.

Figure 8: Vehicle Details Display



Shows details of all available vehicles, including capacity, rate, average speed, and fuel efficiency.

Figure 9.1: Handle a Delivery Request

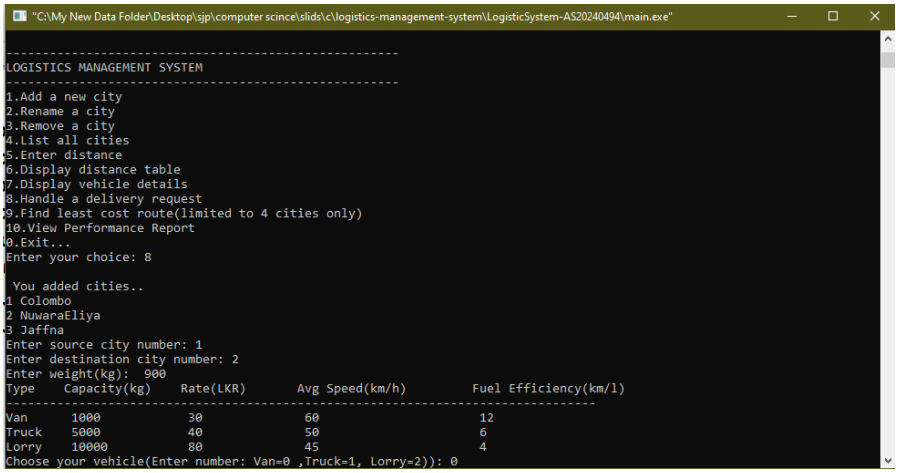
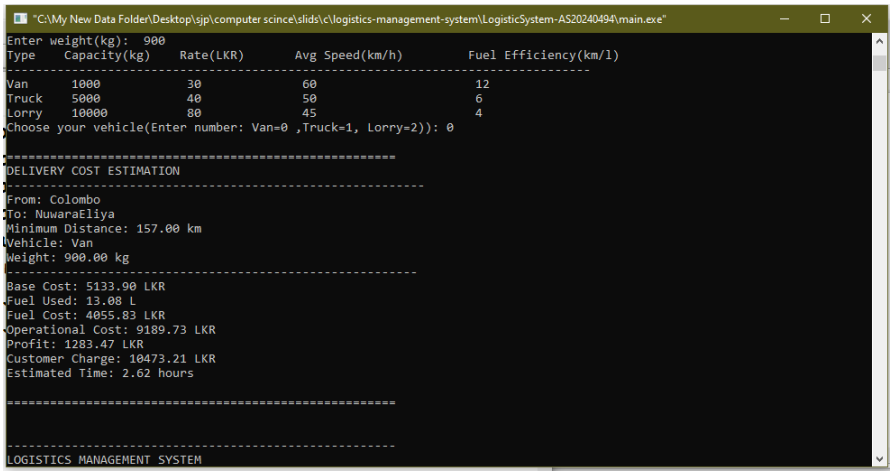
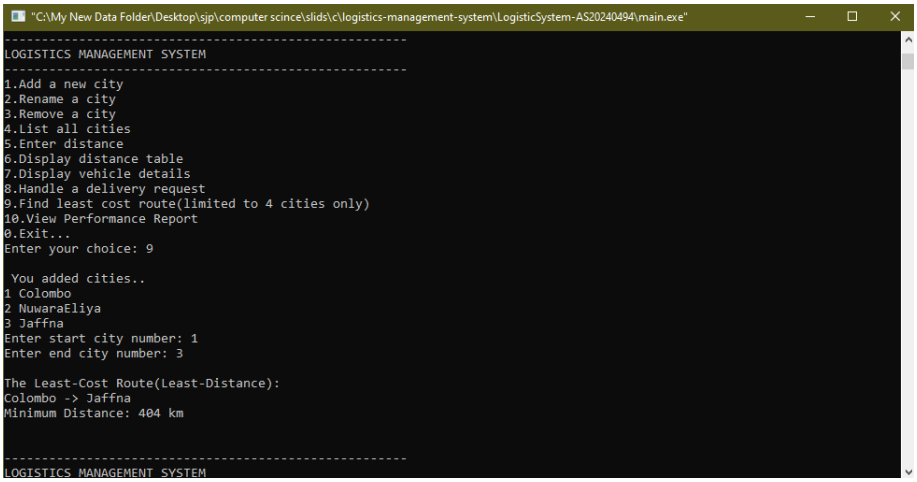


Figure 9.2:



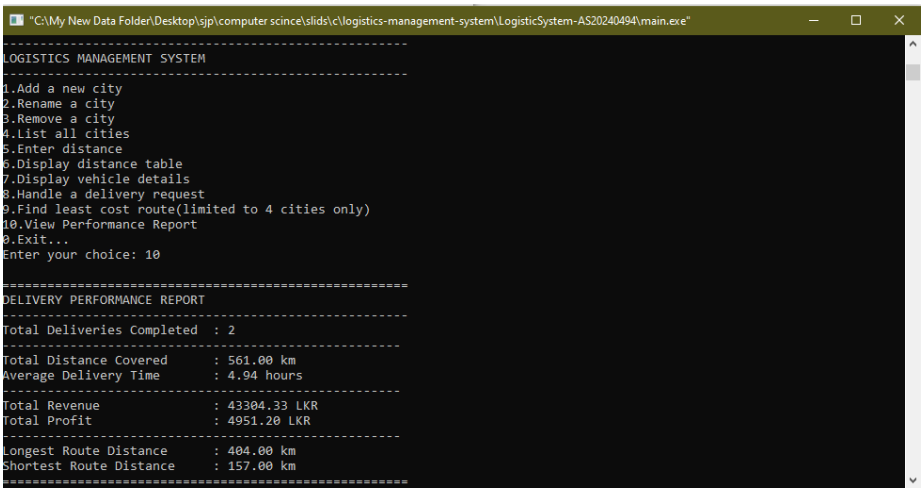
Shows how a delivery request is entered, validated, and processed by the system.

Figure 10: Finding Least Cost Route



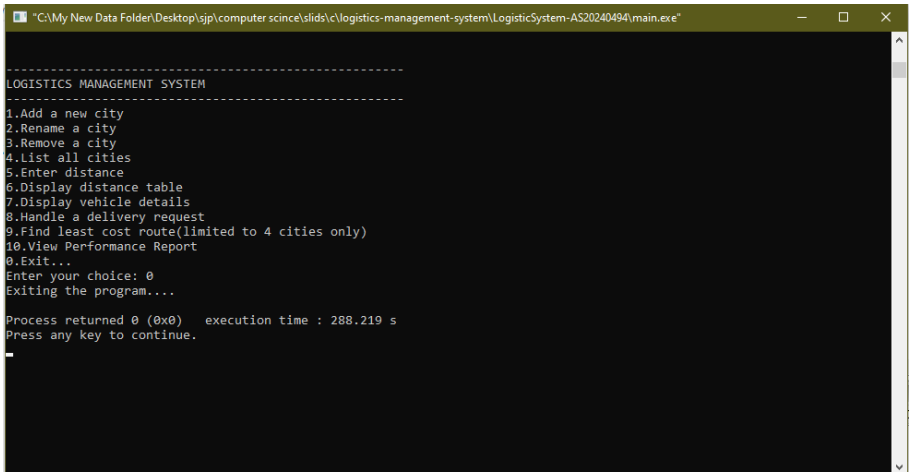
Displays how the system calculates and shows the shortest or least-cost route between two cities.

Figure 11: Report Generation Output



Shows the final performance report, including total deliveries completed, total distance covered, average delivery time, revenue, profit and the longest and Shortest Routes distance.

Figure 12: Exit the program



8. Conclusion

This project successfully simulates a simple logistics management system using C programming concepts. Through this project, I improved my understanding of arrays, functions, conditional statements, and modular programming. The system shows how programming can automate real-world logistics operations efficiently and accurately.