ICT 1011 Computer Programming Assignment (Individual)

University of Sri Jayewardenepura - Faculty of Applied Sciences BSc (General) Degree First Year ICT 1011 Computer Programming Assignment (Individual)

Logistics Management System

Name: D.M.K.H. Dasanayaka

Index number: AS20240946

GitHub repository link: https://github.com/kDash22/projectSem1

Date: 26. 10. 2025

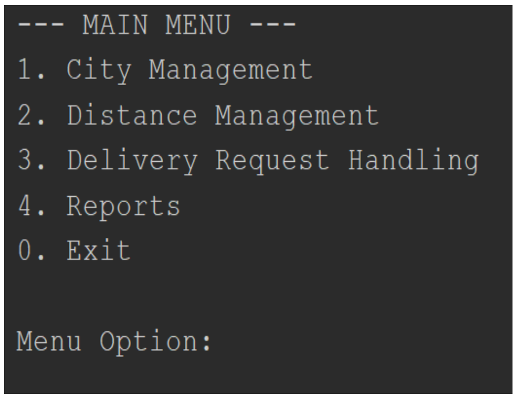
1.Project objective

This project aims to design and implement a menu-driven logistics management system that allows,

- Efficient management of cities and the distances between them
- Placing delivery requests and vehicle selection
- Calculation of delivery costs, fuel usage and operational cost
- Generating delivery receipts and performance reports

2. System Overview

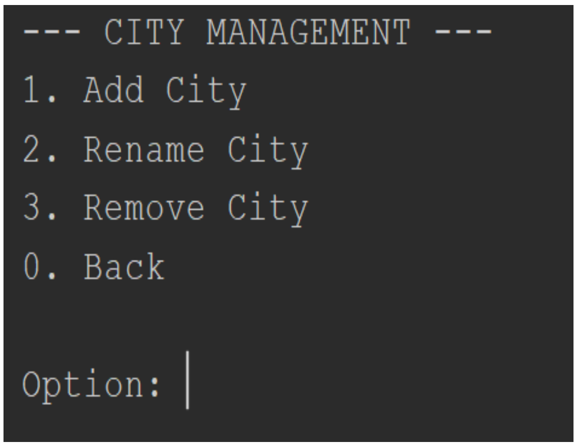
The program is a menu driven application where the user interacts with the program via a series of menus to perform the required tasks.



```
--- MAIN MENU ---
1. City Management
2. Distance Management
3. Delivery Request Handling
4. Reports
0. Exit

Menu Option:
```

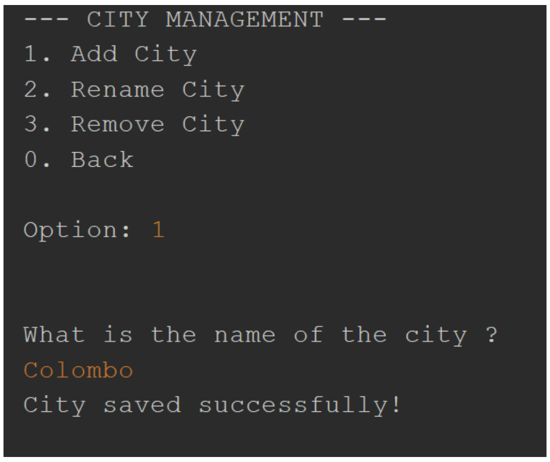
2.1. City Management

This menu option allows the user to add, rename and remove cities.



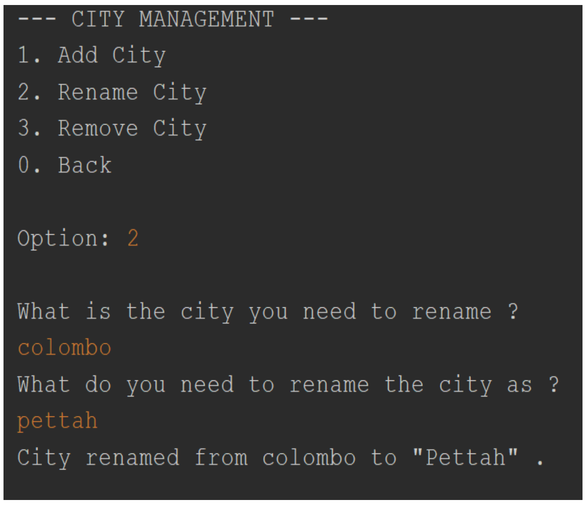
```
--- CITY MANAGEMENT ---
1. Add City
2. Rename City
3. Remove City
0. Back

Option:
```

2.1.1 -> Add city allows the user to add a city into the city database. An index is assigned to the city depending on the cell available in the `cityTable` array. The maximum number of cities in the array is limited to 30 but can be changed by altering the `MAX_CITIES` variable in the program.



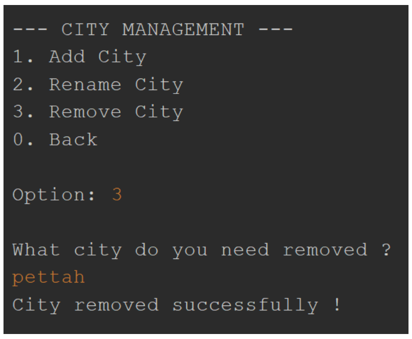
```
--- CITY MANAGEMENT ---
1. Add City
2. Rename City
3. Remove City
0. Back

Option: 1


What is the name of the city ?
Colombo
City saved successfully!
```

2.1.2.-> Rename city allows the user to rename an existing city by entering its previous name and inputting the new name.



```
--- CITY MANAGEMENT ---
1. Add City
2. Rename City
3. Remove City
0. Back

Option: 2

What is the city you need to rename ?
colombo
What do you need to rename the city as ?
pettah
City renamed from colombo to "Pettah" .
```

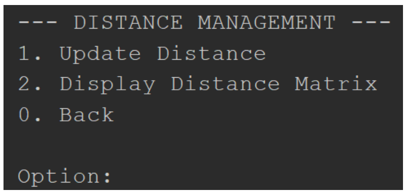
2.1.3.-> Remove city allows the user to remove an existing city by entering its name, this option also removes the relevant distances from the distance menu.



```
--- CITY MANAGEMENT ---
1. Add City
2. Rename City
3. Remove City
0. Back

Option: 3

What city do you need removed ?
pettah
City removed successfully !
```

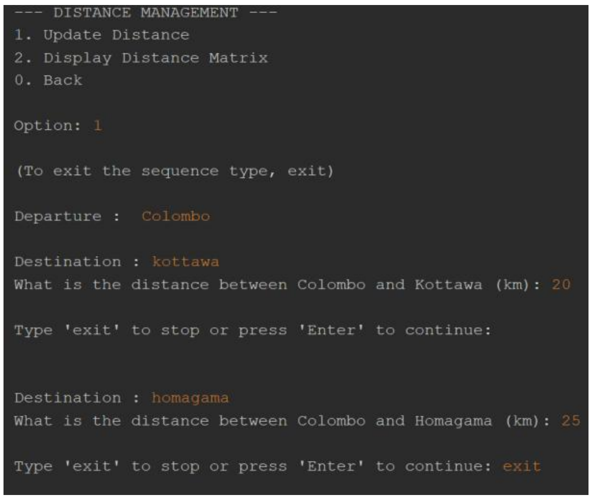
2.2. Distance Management

This menu option allows the user to update the distance matrix and display it.



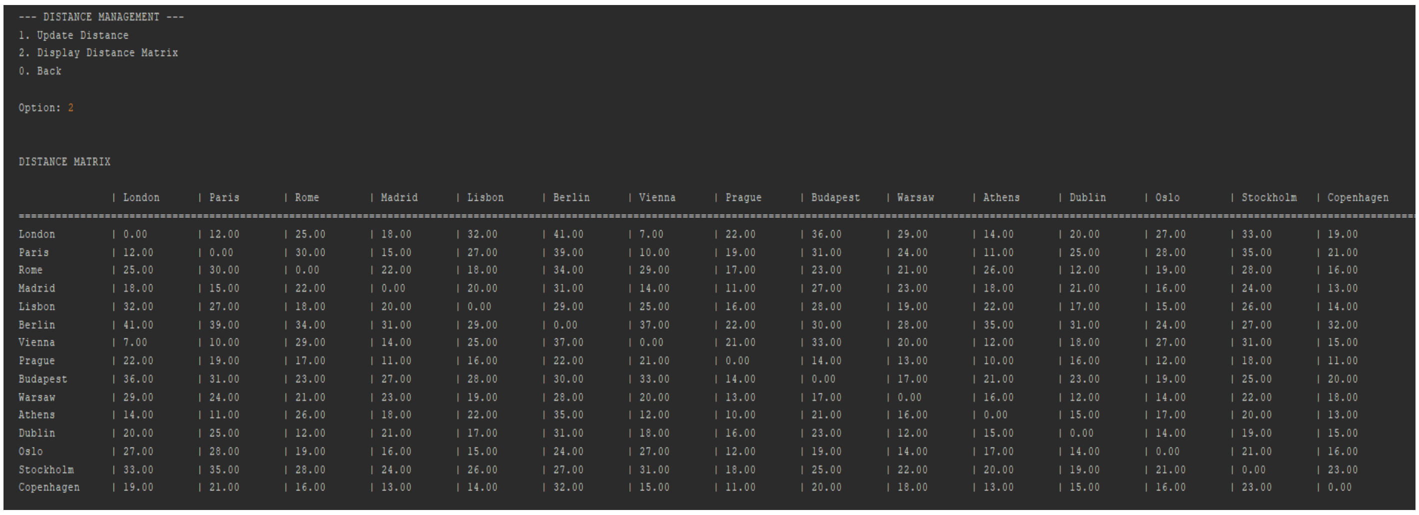
```
--- DISTANCE MANAGEMENT ---
1. Update Distance
2. Display Distance Matrix
0. Back

Option:
```

2.2.1.-> Update Distance option allows the user to update distances, when a departure is chosen it will loop infinitely asking for destination cities until term “exit” is entered. This was done to make bulk entering easier. If not this when a new city is entered the user would’ve had to enter all the combinations one by one. In my scenario by entering the departure city, all the distances to remaining destinations can be added without recalling the option every time.



```
--- DISTANCE MANAGEMENT ---
1. Update Distance
2. Display Distance Matrix
0. Back

Option: 1

(To exit the sequence type, exit)

Departure :  Colombo

Destination : kottawa
What is the distance between Colombo and Kottawa (km): 20

Type 'exit' to stop or press 'Enter' to continue:


Destination : homagama
What is the distance between Colombo and Homagama (km): 25

Type 'exit' to stop or press 'Enter' to continue: exit
```

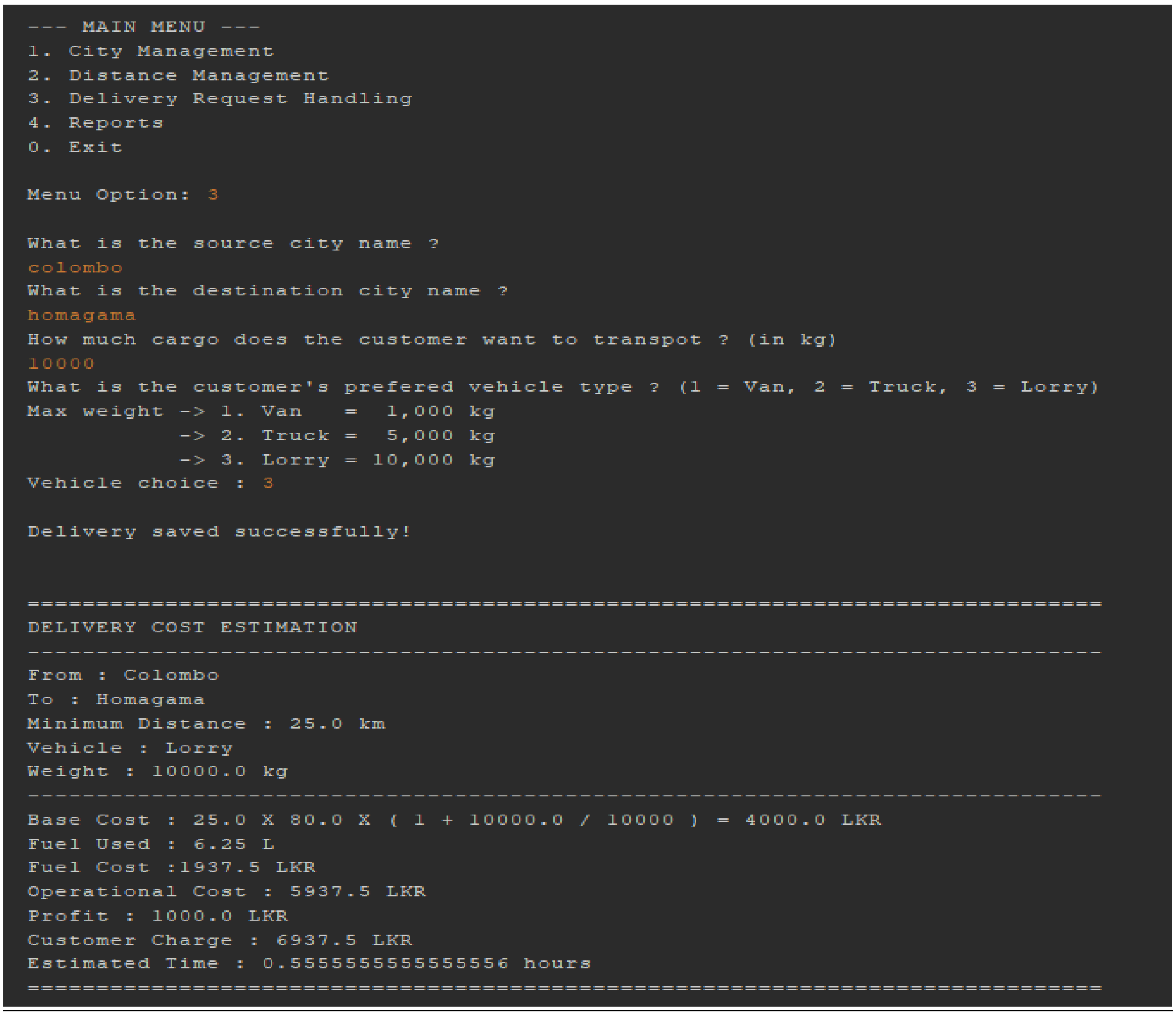
2.2.2.-> Allows the user to display the distance matrix with the relevant cities entered. (Eg: AI generated distance matrix)



```
--- DISTANCE MANAGEMENT ---
1. Update Distance
2. Display Distance Matrix
0. Back

Option: 2


DISTANCE MATRIX

            | London    | Paris     | Rome      | Madrid    | Lisbon    | Berlin    | Vienna    | Prague    | Budapest  | Warsaw    | Athens    | Dublin    | Oslo      | Stockholm | Copenhagen
==================================================================================================================================================================================================
London      | 0.00      | 12.00     | 25.00     | 18.00     | 32.00     | 41.00     | 7.00      | 22.00     | 36.00     | 29.00     | 14.00     | 20.00     | 27.00     | 33.00     | 19.00
Paris       | 12.00     | 0.00      | 30.00     | 15.00     | 27.00     | 39.00     | 10.00     | 19.00     | 31.00     | 24.00     | 11.00     | 25.00     | 28.00     | 35.00     | 21.00
Rome        | 25.00     | 30.00     | 0.00      | 22.00     | 18.00     | 34.00     | 29.00     | 17.00     | 23.00     | 21.00     | 26.00     | 12.00     | 19.00     | 28.00     | 16.00
Madrid      | 18.00     | 15.00     | 22.00     | 0.00      | 20.00     | 31.00     | 14.00     | 11.00     | 27.00     | 23.00     | 18.00     | 21.00     | 16.00     | 24.00     | 13.00
Lisbon      | 32.00     | 27.00     | 18.00     | 20.00     | 0.00      | 29.00     | 25.00     | 16.00     | 28.00     | 19.00     | 22.00     | 17.00     | 15.00     | 26.00     | 14.00
Berlin      | 41.00     | 39.00     | 34.00     | 31.00     | 29.00     | 0.00      | 37.00     | 22.00     | 30.00     | 28.00     | 35.00     | 31.00     | 24.00     | 27.00     | 32.00
Vienna      | 7.00      | 10.00     | 29.00     | 14.00     | 25.00     | 37.00     | 0.00      | 21.00     | 33.00     | 20.00     | 12.00     | 18.00     | 27.00     | 31.00     | 15.00
Prague      | 22.00     | 19.00     | 17.00     | 11.00     | 16.00     | 22.00     | 21.00     | 0.00      | 14.00     | 13.00     | 10.00     | 16.00     | 12.00     | 18.00     | 11.00
Budapest    | 36.00     | 31.00     | 23.00     | 27.00     | 28.00     | 30.00     | 33.00     | 14.00     | 0.00      | 17.00     | 21.00     | 23.00     | 19.00     | 25.00     | 20.00
Warsaw      | 29.00     | 24.00     | 21.00     | 23.00     | 19.00     | 28.00     | 20.00     | 13.00     | 17.00     | 0.00      | 16.00     | 12.00     | 14.00     | 22.00     | 18.00
Athens      | 14.00     | 11.00     | 26.00     | 18.00     | 22.00     | 35.00     | 12.00     | 10.00     | 21.00     | 16.00     | 0.00      | 15.00     | 17.00     | 20.00     | 13.00
Dublin      | 20.00     | 25.00     | 12.00     | 21.00     | 17.00     | 31.00     | 18.00     | 16.00     | 23.00     | 12.00     | 15.00     | 0.00      | 14.00     | 19.00     | 15.00
Oslo        | 27.00     | 28.00     | 19.00     | 16.00     | 15.00     | 24.00     | 27.00     | 12.00     | 19.00     | 14.00     | 17.00     | 14.00     | 0.00      | 21.00     | 16.00
Stockholm   | 33.00     | 35.00     | 28.00     | 24.00     | 26.00     | 27.00     | 31.00     | 18.00     | 25.00     | 22.00     | 20.00     | 19.00     | 21.00     | 0.00      | 23.00
Copenhagen  | 19.00     | 21.00     | 16.00     | 13.00     | 14.00     | 32.00     | 15.00     | 11.00     | 20.00     | 18.00     | 13.00     | 15.00     | 16.00     | 23.00     | 0.00
```

2.3. Delivery Request Handling

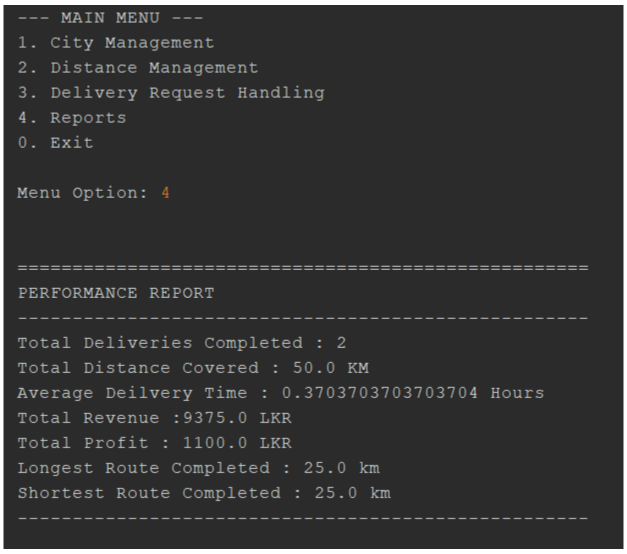
This option allows the user to place a delivery request. The calculations are also done on the backend and the delivery cost estimation receipt is also shown after the order is confirmed. Maximum of 50 orders can be stored in the database for future check ups and for the usage of generating performance reports. The number of maximum orders can be changed by altering the variable `MAX_RECORDS`.



```
--- MAIN MENU ---
1. City Management
2. Distance Management
3. Delivery Request Handling
4. Reports
0. Exit

Menu Option: 3

What is the source city name ?
colombo
What is the destination city name ?
homagama
How much cargo does the customer want to transpot ? (in kg)
10000
What is the customer's prefered vehicle type ? (1 = Van, 2 = Truck, 3 = Lorry)
Max weight -> 1. Van   =  1,000 kg
           -> 2. Truck =  5,000 kg
           -> 3. Lorry = 10,000 kg
Vehicle choice : 3

Delivery saved successfully!


==============================================================================
DELIVERY COST ESTIMATION
------------------------------------------------------------------------------
From : Colombo
To : Homagama
Minimum Distance : 25.0 km
Vehicle : Lorry
Weight : 10000.0 kg
------------------------------------------------------------------------------
Base Cost : 25.0 X 80.0 X ( 1 + 10000.0 / 10000 ) = 4000.0 LKR
Fuel Used : 6.25 L
Fuel Cost :1937.5 LKR
Operational Cost : 5937.5 LKR
Profit : 1000.0 LKR
Customer Charge : 6937.5 LKR
Estimated Time : 0.5555555555555556 hours
==============================================================================
```

2.4. Reports

This option allows the user to get a performance report based on the last 50 deliveries entered, if the delivery record doesn't have 50 records the number of records in the delivery record is used.



```
--- MAIN MENU ---
1. City Management
2. Distance Management
3. Delivery Request Handling
4. Reports
0. Exit

Menu Option: 4


====================================================
PERFORMANCE REPORT
----------------------------------------------------
Total Deliveries Completed : 2
Total Distance Covered : 50.0 KM
Average Deilvery Time : 0.3703703703703704 Hours
Total Revenue :9375.0 LKR
Total Profit : 1100.0 LKR
Longest Route Completed : 25.0 km
Shortest Route Completed : 25.0 km
----------------------------------------------------
```

3.Data Structures

This program uses several different arrays to store the data needed for the program.

- Cities -> String[] cityTable – stores up to 30 city names.
- Distance matrix -> double[][] distanceMatrix – stores distances between cities; -1 indicates undefined.
- Vehicle table ->_final double[][] VEHICLE_TABLE – stores the values needed for calculations regarding the vehicles

Type	Capacity (kg)	Rate per km (LKR)	Avg Speed (km/h)	Fuel Efficiency (km/l)
Van	1000	30	60	12
Truck	5000	40	50	6
Lorry	10000	80	45	4

- Delivery Records: String[][] deliveryRecord – 13-column table for order and cost information; maximum 50 records.

From	To	Minimum Distance(km)	Vehicle	Weight(kg)	Base Cost (LKR)
Colombo	Homagama	25.0	Lorry	10000.0	4000.0

Fuel Used(l)	Fuel Cost (LKR)	Operation Cost (LKR)	Profit (LKR)	Customer Charge (LKR)	Estimated Time (hours)	Rate per km(LKR per km)
6.25	1937.5	5937.5	1000.0	6937.5	0.555556	80.0

- Order record: double[] orderTable – 6 column, 1 row array to temporarily hold order details

Distance	Weight	Rate per km	Vehicle Speed	Efficiency	Fuel Price
orderTable[0]	orderTable[1]	orderTable[2]	orderTable[3]	orderTable[4]	orderTable[5]

4.Methods & Functionalities

Method	Functionality
menu()	Main menu; calls submenus for each module.
addCity()	Adds a new city; capitalizes first letter.
renameCity()	Renames an existing city.
removeCity()	Removes a city and resets its distances.
updateDistance()	Updates distances between two cities.
displayDistanceMatrix()	Prints the distance matrix.
addOder()	Handles new delivery orders and updates delivery records. Prints the delivery cost estimation receipt. (by calling generateDeliveryReciept())
deliveryCost()	Calculates base delivery cost.
estimatedDeliveryTime()	Calculates delivery time based on vehicle speed.
fuelConsumption()	Calculates fuel used for a delivery.
fuelCost()	Calculates total fuel cost.
TotalOperationCost()	Calculates total operational cost including fuel and base cost.
profit()	Calculates profit (25% of base cost).
customerCharge()	Calculates total amount to charge customer.
minimumDistance()	Approximates shortest route using up to 4 intermediary cities.
generateDeliveryReciept()	Prints formatted delivery receipt for an order.
generatePerformanceReport()	Summarizes all deliveries, revenue, and profits.
storeCityTable() / storeDistanceMatrix() / storeDeliveryRecord()	Save data to .txt files.
loadCityTable() / loadDistanceMatrix() / loadDeliveryRecord()	Load data from .txt files.

5. Algorithms

5.1 Minimum Distance Calculation

Uses nested loops to calculate the minimum distance up to 4 intermediary cities to ensure the usage of a shorter distance for calculations rather than just the direct distance.

5.2 Delivery Cost Calculations

Note: for calculations the distance is taken from the minimum distance calculation.

- Base Cost = Distance × Rate × (1 + Weight / 10000)
- Fuel Used = Distance / Vehicle Efficiency
- Fuel Cost = Fuel Used × Fuel Price
- Total Operation Cost = Base Cost + Fuel Cost
- Profit = Base Cost × 0.25
- Customer Charge = Total Operation Cost + Profit
- Estimated Delivery Time = Distance / Vehicle Speed

6. File Handling

File handling is an important process done by the java program as it reads the relevant text files to update the data structures in the java program making sure the data is not lost after exiting the program. The said text files are updated as soon as a change happens to a data structure in the runtime of the program.

storeCityTable() / storeDistanceMatrix() / storeDeliveryRecord()	Save data to .txt files.
loadCityTable() / loadDistanceMatrix() / loadDeliveryRecord()	Load data from .txt files.

Above methods are used in the program to store and load the data.

7.Assumptions and Limitations

- Maximum 30 cites and 50 delivery records, although can be altered easily in the program by changing the variables MAX_CITIES and MAX_RECORDS respectively.
- Minimum distance calculation only uses a combination of 4 cities.
- Vehicle capacities: Van = 1000 kg, Truck = 5000 kg, Lorry = 10000 kg.
- The fuel price is fixed at 310LKR/L, different types of fuel usage is not taken into account.
- All user inputs are validated for correct data type and range.
- A static final double INFINITY = 999999999; is used to represent an extremely large value in methods that require comparison or initialization with a high upper bound (e.g., shortest path calculations)
- A single global Scanner object (input) is used throughout the program to handle all user inputs efficiently and prevent redundant object creation.

8.Conclusion

- The program successfully manages the logistics operations and meets the functional requirements of the system.
- Some. Txt files used in testing are also present in the project folder.