DMRC Metro Simulator – README

(Bonus Component)

Submitted By:Satwik Singh (2025461)

The program achieves enhanced functionality through three major extensions: advanced multi-transfer routing, seamless line expansion, and a time-based fare calculation system.

1. Advanced Routing: Multi-Interchange Paths (Two Transfers)

The core function responsible for this extension is twointer(src, dest, startmin). This feature moves beyond simple single-line and one-interchange travel to solve complex routes requiring a maximum of two transfers and three distinct metro lines.

Search and Optimization Logic

- Iterative Line Search**:** The function first identifies all available lines (**L1**) serving the source station and all lines (L3) serving the destination station. It then iterates through *every other line* (L2) in the entire network that is neither L1 nor L3. This ensures all possible three-line combinations are tested.
- Interchange Candidate Identification**:** For a given combination of (L1, L2, L3), the function identifies candidate stations:
    - Interchange A**:** Stations that serve both L1 and L2.
    - Interchange B**:** Stations that serve both L2 and L3.
- Route Evaluation (Three Legs): For every combination of Interchange A and Interchange B, the complete journey is calculated in three sequential steps:
    - Leg 1 (L1): singleLine(L1, src, A, startmin) calculates the arrival time at the first transfer station (Arr1).
    - Transfer 1 Delay: A fixed 3-minute delay is added to Arr1 to get the departure time for the second leg (Depart2).
    - Leg 2 (L2): singleLine(L2, A, B, Depart2) calculates the arrival time at the second transfer station (Arr2).
    - Transfer 2 Delay: A second fixed 3-minute delay is added to Arr2 to get the departure time for the third leg (Depart3).
    - Leg 3 (L3): singleLine(L3, B, dest, Depart3) calculates the Final Arrival Time.

- Best Path Selection: The function maintains track of the overall minimum Final Arrival Time found across all L1/L2/L3 combinations and all Interchange A/Interchange B pairs.

Integration with planJourney

The main routing function, planJourney, considers the result from twointer as just one scenario. It compares the best arrival time from the Two-Interchange route with the best times from the Single Line and One-Interchange routes to guarantee that the final itinerary displayed to the user is the absolute fastest available route, regardless of the number of transfers (0, 1, or 2).

2. Line Expansion: Red and Orange Lines

The program's design, which models the network as an interconnected graph, allows for easy expansion without modifying the core routing algorithms.

Data Integration

- metro_data.txt Parsing: When the program loads data, records tagged with "Red" or "Orange" are processed just like "Blue" or "Magenta."
- Graph Population:
    - travelltime: Stores the segment-specific travel times for the new lines.
    - map: Creates new lists (map["red"], map["orange"]) maintaining the station order for each line.
    - interchanges: Automatically updates to include any station served by both a new line and an existing line (e.g., "Dwarka Sector 21" now connecting "blue" and "orange").

Dynamic Support

Because the core functions like cumtime (calculates time between two stations on a single line) and nextstattrain (calculates train departure/arrival considering frequency) operate generically based on the line key, they instantly support the new Red and Orange lines as soon as their data is loaded. These new lines can therefore be used seamlessly in the single-line path, the first leg of a transfer, or the final leg of a three-line journey.

3. Fare Calculation Module

The fare system is implemented through the fare(totmins) function and provides a simplified, time-based cost structure.

Time-Based Model

The critical design choice here is the assumption that the fare is proportional to the total time spent on the metro system (totmins), which includes waiting time, travel time, and transfer delays. This is determined by the formula:

Total Time in Minutes = Final Arrival Time (in min) - \Initial Start Time (in min)

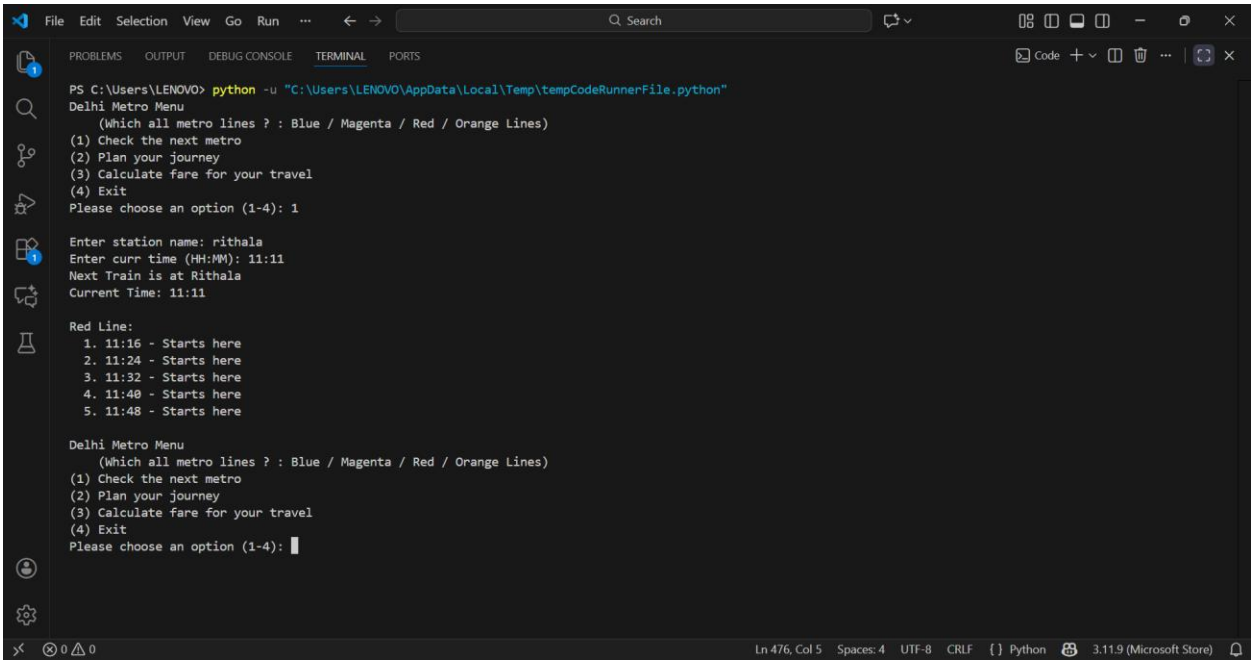
Fare Structure Logic

- The fare(totmins) function uses conditional logic (if/elif/else) to assign a fixed fare amount based on which time slab the total journey duration falls into.
    - 0–10 min : ₹10
    - 11–20 min : ₹20
    - 21–30 min : ₹30
    - 31–45 min : ₹40
    - 46 and above min :₹50

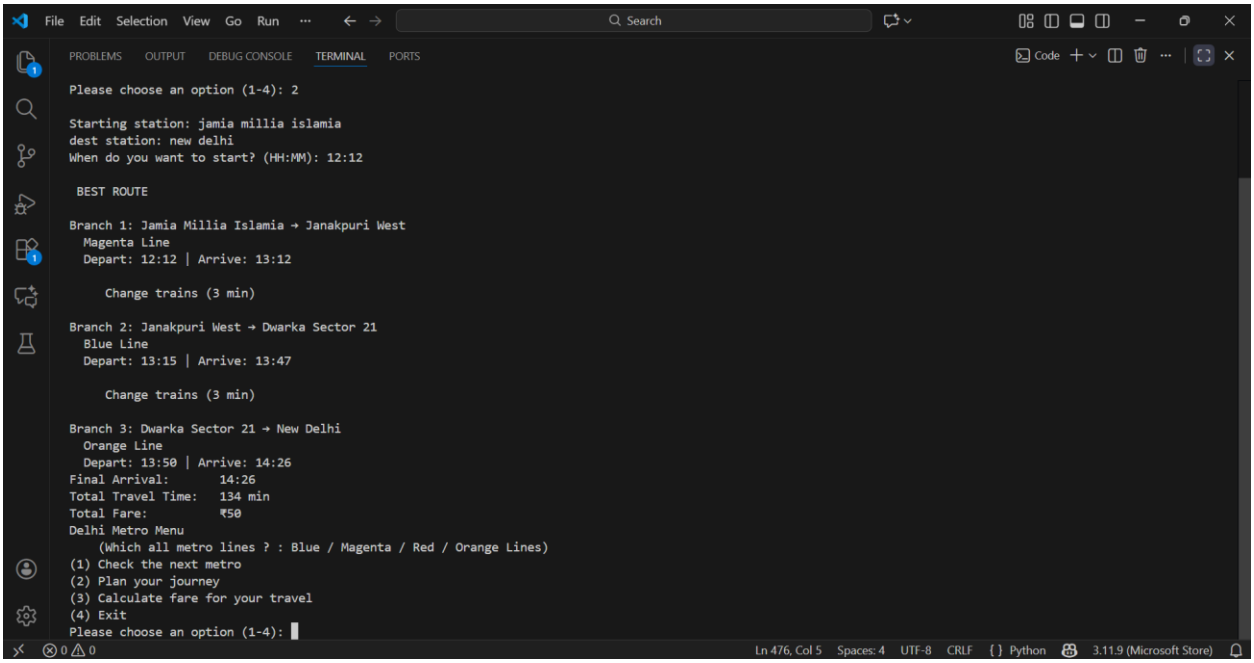
This simple, stepped function satisfies the bonus requirement to include fare calculation and ensures that every itinerary generated by planJourney is complete with an expected cost.

Screenshots of the bonus part:

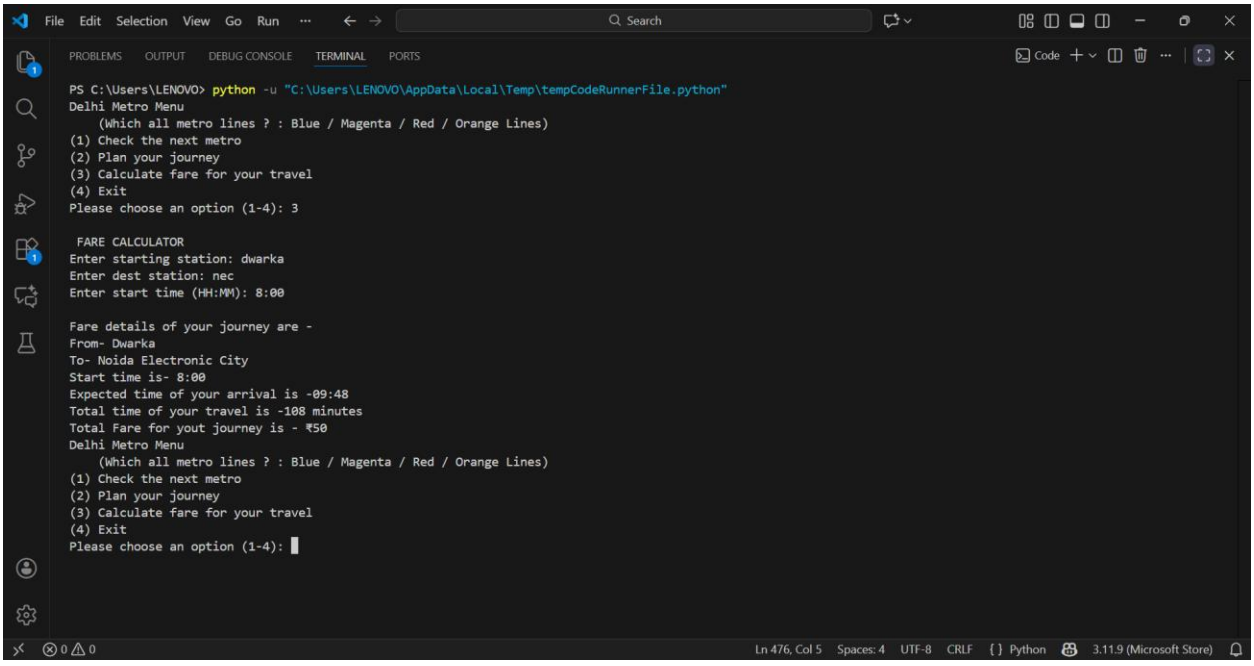
1) Chose 1, source station: Rithala, shows the time of the next train, since it is the start so every train starts at interval of 8 minutes(since time entered is 11:11(non peak hours))



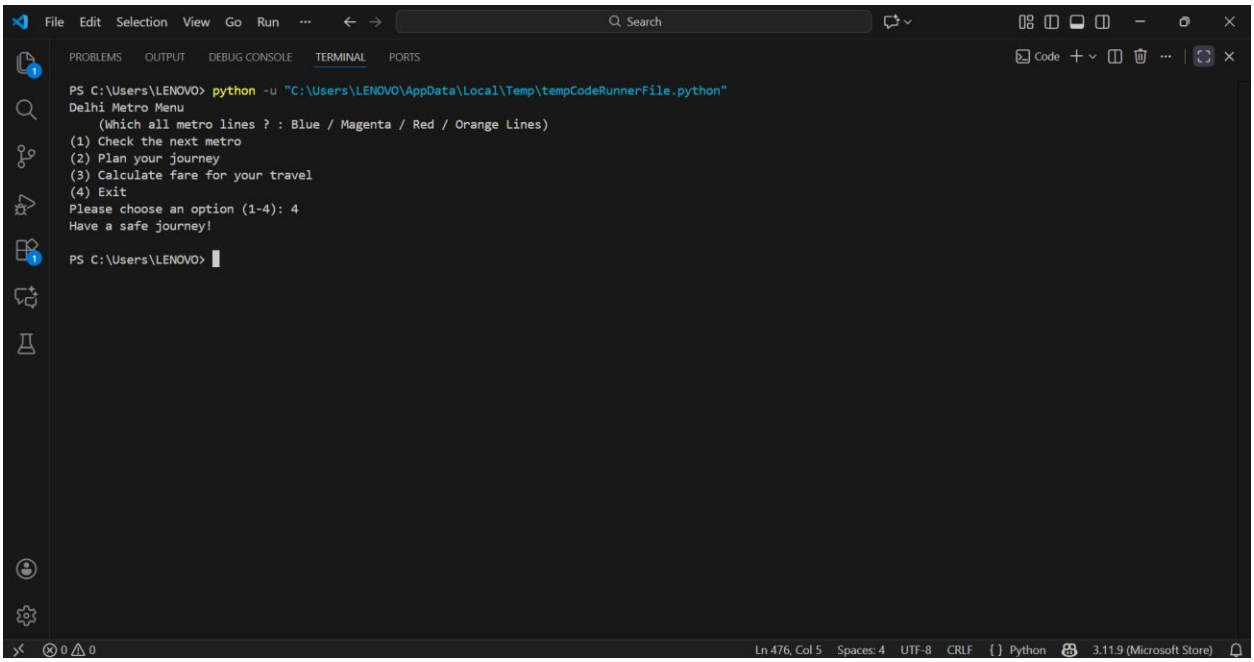
2) Chose 2, source: Jamia Millia Islamia(magenta line) , Destination: New Delhi(Orange line), there will be 2 interchanges and 3 lines required , start from Jamia Millia Islamia station (magenta line) till Janakpuri West (interstation for blue and magenta line), take blue line towards Dwarka Sector 21 and go till Dwarka sector 21 (station interchange for orange and blue line), then take orange line till New Delhi.



3) Fare details, source: Dwarka, destination: Noida Electronic City(used nec as allowed in this program),calculates time which comes out to be 108 mins and as per the scheme the charges will be 50 rupees.



4) Exit the DMRC Simulator: choose 4


File Edit Selection View Go Run
Search
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Code
PS C:\Users\LENOVO> python -u "C:\Users\LENOVO\AppData\Local\Temp\tempCodeRunnerFile.python"
Delhi Metro Menu
(Which all metro lines ? : Blue / Magenta / Red / Orange Lines)
(1) Check the next metro
(2) Plan your journey
(3) Calculate fare for your travel
(4) Exit
Please choose an option (1-4): 4
Have a safe journey!
PS C:\Users\LENOVO>
Ln 476, Col 5 Spaces: 4 UTF-8 CRLF { } Python 3.11.9 (Microsoft Store)