DMRC Metro Simulator – README

(Main Assignment)

Submitted By:Satwik Singh (2025461)

1. Data Sources

The data has been collected from the official app of Delhi Metro Sarthi Mobile Application.

2. Assumptions Made

- Interchange time between any two lines is fixed at 3 minutes although it may take more time in real world.
- The station entered by the user are same as that present on the Delhi Metro Sarthi App except the following stations:

```
correction = {
    "noida e city": "noida electronic city",
    "nec": "noida electronic city",
    "electronic city": "noida electronic city",
    "bg": "botanical garden",
    "botanical": "botanical garden"
}
```

- The time taken between stations has although been taken from the Delhi Metro Sarthi App but may have certain shift from the actual time in certain places.

3. Special Features of the Program

- Supported lines: Blue and Magenta lines(Main) with Red and Orange lines(Bonus part)
- Special handling for Blue Line branches:
    - Blue Vaishali and Blue Noida are handled separately.
    - "Yamuna Bank" acts as the branch-diverging interchange.

- Station names are case-insensitive but does not caters the spacing if the station name has more than one word.
- Common mis-typed inputs are auto-corrected (e.g., nec : Noida Electronic City).
- The program shows:
    - Best route
    - Interchange points
    - Arrival time
    - Total fare
- Dynamic frequency:
    - Peak hours (8–10 AM & 5–7 PM) : trains starts from each terminal every 4 minutes
    - Non-peak hours :trains start from each terminal at every 8 minutes

4. Instructions to Run the Program

Step 1: Prepare metro_data.txt

- Save the metro_data.txt file on your Desktop.

Step 2: Save the Python Script

- Save the provided Python code as: metro_simulator.py

Step 3: Run the Program

Open Command Prompt / Terminal and run: python metro_simulator.py

Step 4: Use the Menu Options (Menu Driven)

You will see:

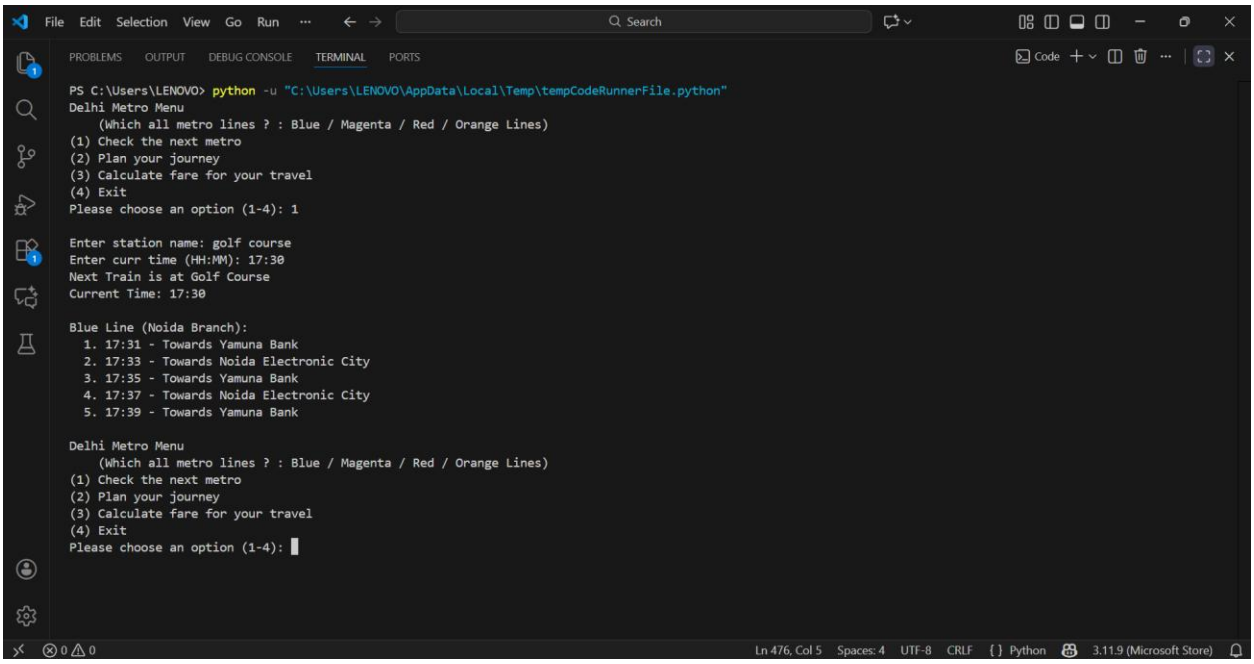
1) Next Metro
2) Journey Planner
3) Fare Calculator
4) Exit

Option 1: Next Metro

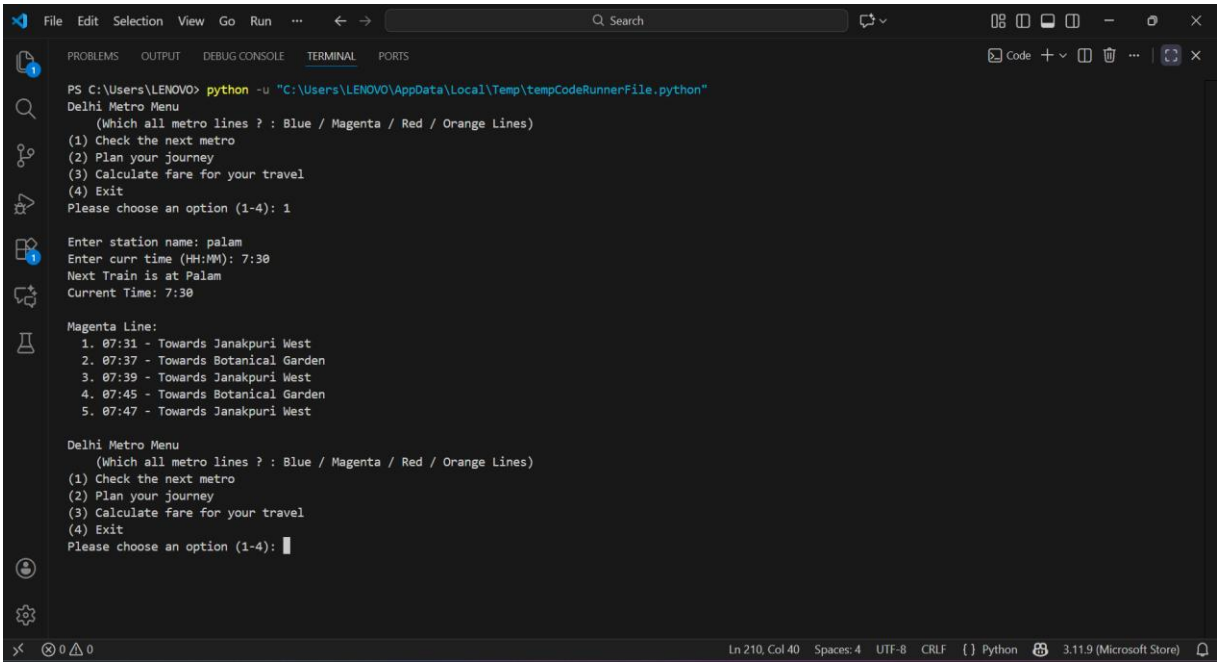
- Enter station name
- Enter time in HH:MM format
- Shows next 5 arriving metros at a given station

Shows the arrival time of the next train at the entered station towards both the direction of the line with respect to the station in both the directions

- This screenshot shows when option 1 is chosen, source is Golf Course and time is 17:30(peak hours), gives the next 5 trains towards both side Noida Electronic City and Yamuna Bank.



- Source: Palam, Current Time: 7:30, shows the next 5 metro trains towards both terminals of magenta line (Botanical Garden and Janakpuri West)

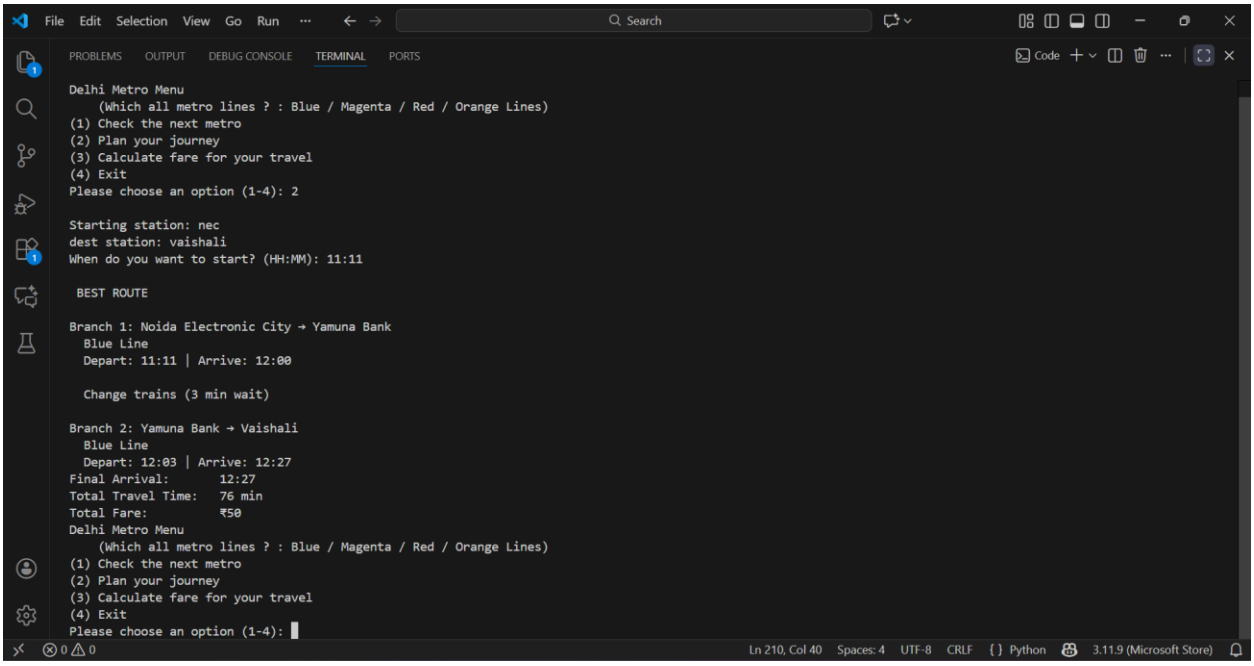


Option 2: Journey Planner

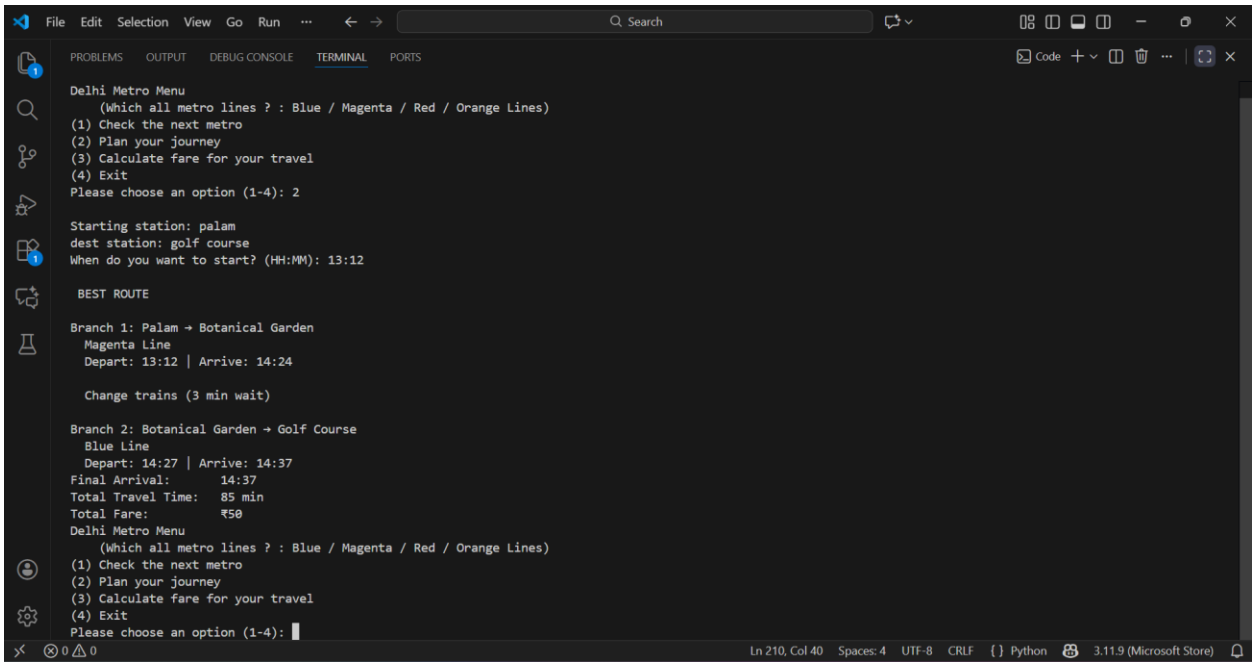
- Enter source station
- Enter destination station
- Enter start time (HH:MM)
- Shows best route, interchanges, timings & fare

Shows the total time taken for the journey including fare and the change time required at any interchange station(fixed at 3 mins for this program)

- Source: Noida Electronic City, Destination: Vaishali, Current time: 11:11, show the route with an interchange required at Yamuna Bank with an assumed delay of 3 mins for interchange.

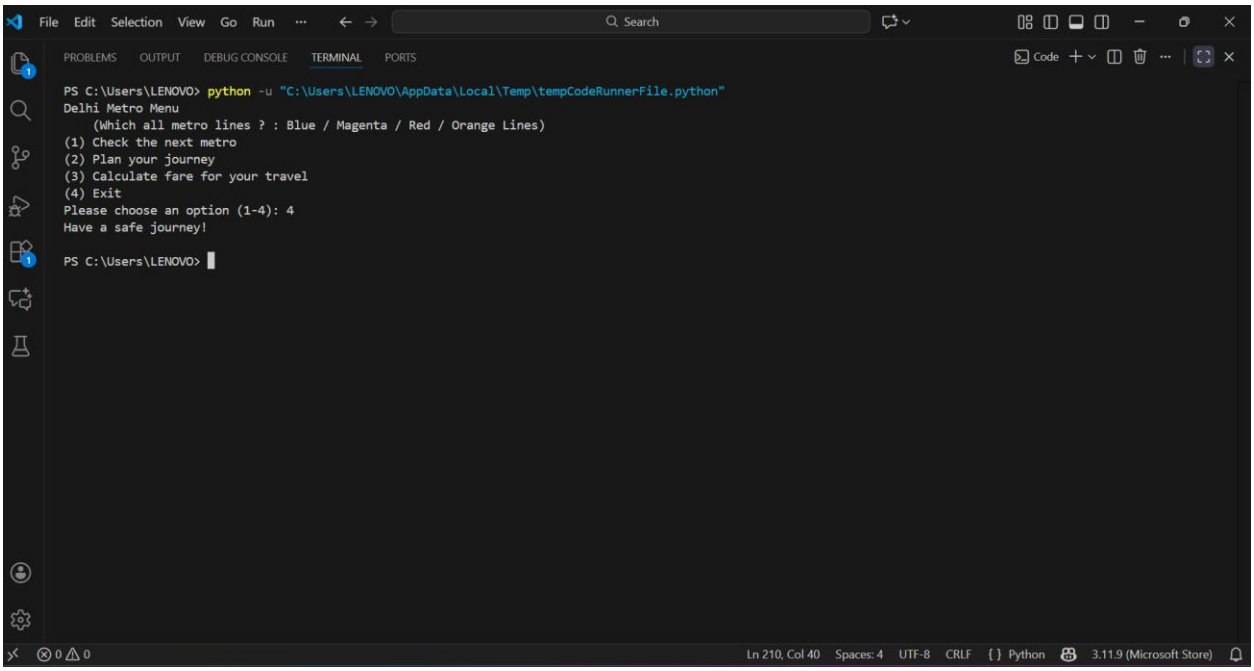


- Source: Palam, Destination: Golf Course, Current time:13:12, show the journey planner with interchange required at Botanical Garden(interstation between blue line and magenta line)



Option 4: Exit

- Closes the simulator



5) WORKING OF THE CODE

The Python program simulates the Delhi Metro system, specifically focusing on the Blue and Magenta lines (with extensions for Red and Orange lines as well). The core working principle of the simulator is to model the metro network as a graph, where stations are nodes and the travel time between adjacent stations forms the weights. It uses a time-based simulation approach, converting all clock times (HH:MM) into minutes past midnight for easy arithmetic, and calculates train arrivals based on a fixed schedule and defined operating frequencies.

1. Data Initialization and Network Model

Data Loading

The initial block reads the metro_data.txt file (which is assumed to be present at the specified path). It parses the raw route data into a list of dictionaries called metdata. This list contains details about each segment: the line, source station, next station, travel time, and whether the source is an interchange.

Core Network Structures

The data is then organized into several global dictionaries essential for routing and timing:

- map: This dictionary stores the ordered sequence of stations for every line and branch (e.g., blue_vaishali, blue_noida, magenta). The order is crucial for calculating cumulative travel time along a continuous path.
- travelltime: This acts as the adjacency information, storing the time in minutes required to travel between any two adjacent stations on a specific line, covering both forward and reverse directions.
- statline: This maps every station to all metro lines that serve it. This is used to determine which lines a traveler can use from a starting station.
- interchanges: This lists all stations where a transfer between two different metro lines is possible, based on stations being served by multiple lines.

2. Time and Frequency Logic

Time Conversion

The convert(t) function changes a time string like "08:22" into the total number of minutes past midnight (e.g., 8 * 60 + 22 = 502 minutes). The reverseconv(m) function does the reverse. This makes time arithmetic simple and avoids complex datetime objects.

Service Hours

The service runs from 06:00 AM (sercstarts = 360 min) to 11:00 PM (serends = 1380 min).

Frequency Calculation

The freqq(minute) function determines the train frequency:

- 4 minutes during Peak Hours (8:00–10:00 AM and 5:00–7:00 PM).
- 8 minutes during Off-Peak Hours.

3. Metro Timings Module (Option 1)

The trainarrival function simulates the arrival schedule:

- It iterates through both end terminals of the specified line/branch.
- For a given terminal, it calculates the cumulative travel time (travel) from that terminal to the target station using cumtime.
- It then simulates train departures from the terminal, starting at 06:00 AM and incrementing by the current frequency.
- The arrival time at the station is (Terminal Departure Time) +(Travel Time)
- It lists all calculated arrival times that are after the user's current time and before service ends.
- The nxtmet function calls this logic for all lines serving the station and displays the next five predicted arrival times for the user.

4. Journey Planning Module (Option 2)

The planning functions utilize a shortest-path approach by calculating the total travel time for various scenarios and choosing the fastest one.

Cumulative Travel (cumtime)

This is the fundamental routing function. It takes two stations and a line/branch, finds their positions in the ordered map list, and sums the travel times between consecutive stations to find the direct station-to-station travel time.

Next Train Departure (nextstattrain)

This function calculates the time a traveler will actually depart the source station.

- It calculates the travel time from the line's terminal to the source station.
- It simulates terminal departures (starting at 06:00 AM).

- It finds the first terminal departure such that Arrival at Source = Terminal Departure + Time to Source is greater than or equal to the traveler's desired starttime.
- The traveler then boards this train. The final arrival time is Arrival at Source + Time from Source to Destination.

Route Scenarios

1. Single Line **(**singleLine**)**: Directly checks if a single line connects the source and destination using nextstattrain.
2. One Interchange **(**interchange**)**:
    - Finds the fastest route via any single transfer station.
    - Calculates Branch 1 Arrival (Source -> Interchange).
    - Adds a 3-minute interchange delay.
    - Calculates Branch 2 Arrival (Interchange ->Destination) starting from the delayed time.
    - The route with the minimum final arrival time is selected.
3. Two Interchanges (twointer): This is an extension that searches for routes involving three distinct lines and two interchange stations (A and B). It calculates the arrival time for three consecutive legs, including two 3-minute delays.

Final Output

The planJourney function finds the best route among all scenarios (single, one-interchange, two-interchange), determines the Total Travel Time (Final Arrival - Initial Start Time), calculates the Fare using the fare function, and prints the detailed itinerary.

6. User Experience

- The program includes input error handling for time formats and station names. It uses the mistofuser function to correct common misspellings or abbreviations (e.g., "BG" to "botanical garden") to improve user experience.
- Menu driven program so that the user doesn’t have to search again and again which function is to be run and only get the desired information.