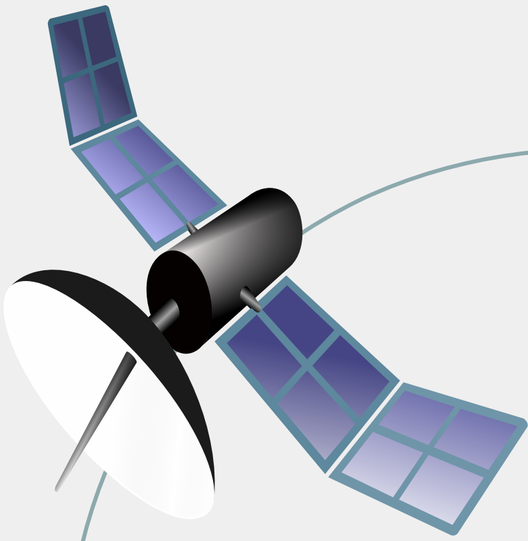
GPS TOLL BASED SYSTEM SIMULATION USING PYTHON



SOLUTION

```
import random
import pandas as pd
from datetime import datetime, timedelta

def generate_gps_data(num_points, lat_range, lon_range):
    data = []
    start_time = datetime.now()
    for _ in range(num_points):
        lat = random.uniform(lat_range[0], lat_range[1])
        lon = random.uniform(lon_range[0], lon_range[1])
        timestamp = start_time + timedelta(seconds=random.randint(1, 60))
        data.append((timestamp, lat, lon))
    return pd.DataFrame(data, columns=['timestamp', 'latitude', 'longitude'])

# Example usage
gps_data = generate_gps_data(10, (40.7128, 40.7486), (-74.0060, -73.9352))
print(gps_data)
from geopy.distance import geodesic

def calculate_total_distance(gps_data):
    total_distance = 0.0
    for i in range(1, len(gps_data)):
        coords_1 = (gps_data.iloc[i-1]['latitude'], gps_data.iloc[i-1]['longitude'])
```

✓ 1s completed at 18:58

```
def calculate_total_distance(gps_data):
    total_distance = 0.0
    for i in range(1, len(gps_data)):
        coords_1 = (gps_data.iloc[i-1]['latitude'], gps_data.iloc[i-1]['longitude'])
        coords_2 = (gps_data.iloc[i]['latitude'], gps_data.iloc[i]['longitude'])
        total_distance += geodesic(coords_1, coords_2).kilometers
    return total_distance

# Example usage
total_distance = calculate_total_distance(gps_data)
print(f"Total Distance: {total_distance} km")
def calculate_toll(distance, rate_per_km=50):
    return distance * rate_per_km

# Example usage
toll = calculate_toll(total_distance)
print(f"Toll Charge: ruprees {toll:.2f}")
def simulate_multiple_vehicles(num_vehicles, num_points, lat_range, lon_range):
    results = []
    for vehicle_id in range(num_vehicles):
        gps_data = generate_gps_data(num_points, lat_range, lon_range)
        total_distance = calculate_total_distance(gps_data)
        toll = calculate_toll(total_distance)
        results.append((vehicle_id, total_distance, toll))
```

```
    return total_distance

# Example usage
total_distance = calculate_total_distance(gps_data)
print(f"Total Distance: {total_distance} km")
def calculate_toll(distance, rate_per_km=50):
    return distance * rate_per_km

# Example usage
toll = calculate_toll(total_distance)
print(f"Toll Charge: ruprees {toll:.2f}")
def simulate_multiple_vehicles(num_vehicles, num_points, lat_range, lon_range):
    results = []
    for vehicle_id in range(num_vehicles):
        gps_data = generate_gps_data(num_points, lat_range, lon_range)
        total_distance = calculate_total_distance(gps_data)
        toll = calculate_toll(total_distance)
        results.append((vehicle_id, total_distance, toll))
    return pd.DataFrame(results, columns=['vehicle_id', 'total_distance', 'toll_charge'])

# Example usage
simulation_results = simulate_multiple_vehicles(5, 10, (40.7128, 40.7486), (-74.0060, -73.9352))
print('simulation results:', simulation_results)
```

OUTPUT

```
                        timestamp   latitude  longitude
0 2024-07-13 13:28:24.887806  40.734013 -73.997045
1 2024-07-13 13:28:17.887806  40.729783 -73.991091
2 2024-07-13 13:28:30.887806  40.739193 -73.991585
3 2024-07-13 13:28:31.887806  40.732344 -73.988028
4 2024-07-13 13:28:47.887806  40.747743 -73.996519
5 2024-07-13 13:28:59.887806  40.741863 -73.970816
6 2024-07-13 13:28:40.887806  40.729337 -73.998544
7 2024-07-13 13:28:46.887806  40.739359 -73.947559
8 2024-07-13 13:28:38.887806  40.733468 -73.954462
9 2024-07-13 13:28:48.887806  40.742078 -74.001941
Total Distance: 18.844628521256443 km
Toll Charge: ruprees 942.23
simulation results:    vehicle_id  total_distance  toll_charge
0              0       26.924594  1346.229679
1              1       28.322153  1416.107627
2              2       23.428516  1171.425789
3              3       20.040056  1002.002783
4              4       20.986151  1049.307565
```

FEATURES OFFERED

- 1. GPS Data Generation: Generates random GPS data points with timestamps, latitude, and longitude within a specified range.
-
- 2. Total Distance Calculation: Calculates the total distance traveled by a vehicle based on the generated GPS data points.
-
- 3. Toll Charge Calculation: Calculates the toll charge based on the total distance traveled and a specified rate per kilometer.
-
- 4. Multi-Vehicle Simulation: Simulates GPS data generation, total distance calculation, and toll charge calculation for multiple vehicles.
-

- 5. Data Storage: Stores the generated GPS data, total distance, and toll charge for each vehicle in a pandas DataFrame.
-
- 6. Randomized Timestamps: Generates random timestamps for each GPS data point, simulating real-world GPS data collection.
-
- 7. Geodesic Distance Calculation: Uses the geodesic distance formula to calculate the distance between two GPS coordinates, providing an accurate measurement of distance traveled.
-
- 8. Customizable Parameters: Allows users to specify the number of GPS data points, latitude and longitude ranges, and toll rate per kilometer.
-
- 9. Data Analysis: Enables analysis of the generated data, including total distance traveled and toll charges for each vehicle.

PROCESS FLOW

- Generate_GPS_data

1. Start
2. Initialize data list
3. Set start time to current datetime
4. Loop num_points times
   - Generate random lat and lon within ranges
   - Calculate timestamp with random offset
   - Append (timestamp, lat, lon) to data list
5. Return DataFrame from data list

- Calculate_total_distance

1. Start
2. Initialize total_distance to 0
3. Loop through gps_data rows (starting from 2nd row)
   - Calculate geodesic distance between current and previous points
   - Add distance to total_distance
4. Return total_distance

- Calculate_toll

1. Start
2. Return distance * rate_per_km

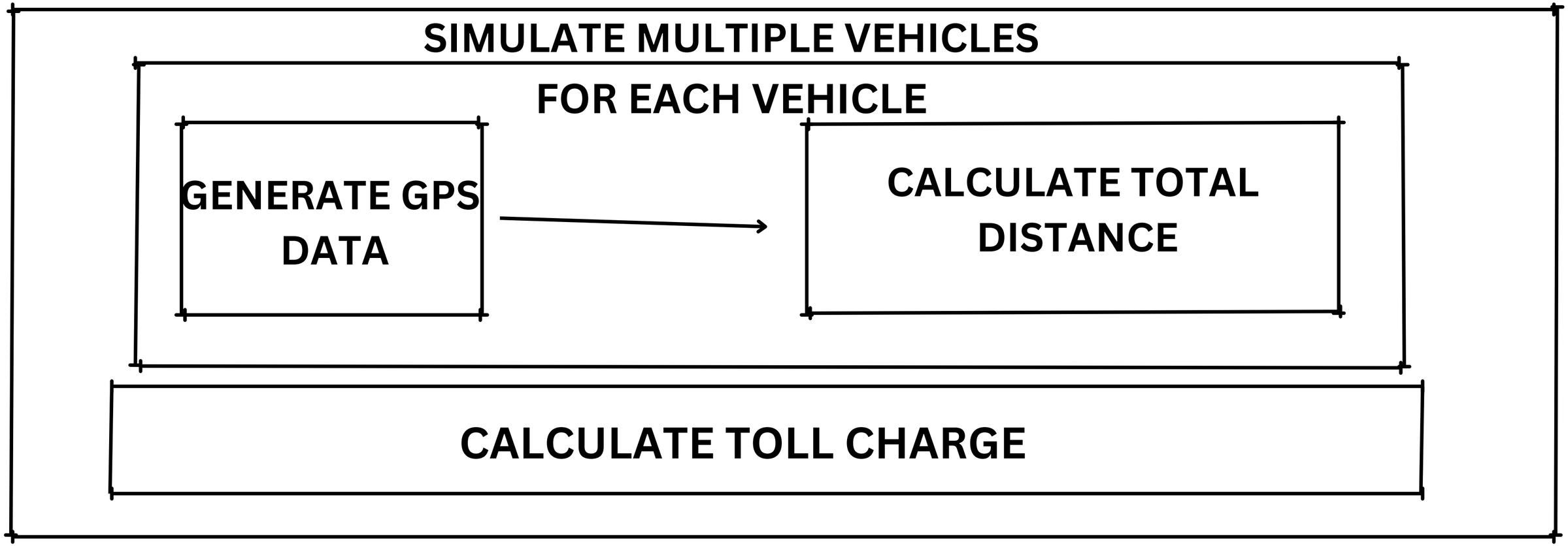
- simulate_multiple_vehicles

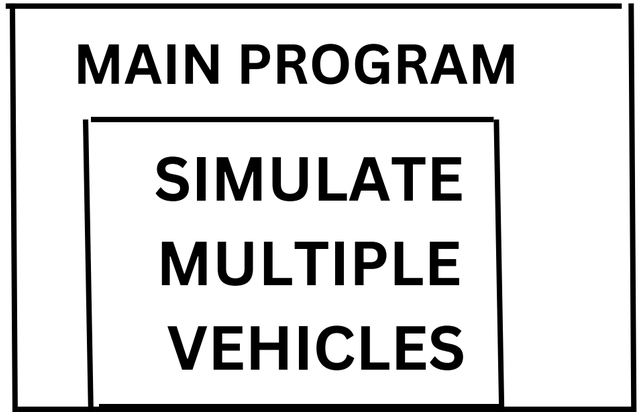
1. Start
2. Initialize results list
3. Loop num_vehicles times
   - Generate gps_data for current vehicle
   - Calculate total_distance for current vehicle
   - Calculate toll for current vehicle
   - Append (vehicle_id, total_distance, toll) to results list
4. Return DataFrame from results list

- Main Program

1. Start
2. Generate gps_data for 10 points
3. Calculate total_distance for gps_data
4. Calculate toll for total_distance
5. Simulate multiple vehicles (5 vehicles, 10 points each)
6. Print results

ARCHITECTURE DIAGRAM





TECHNOLOGIES USED

1. Python: The program is written in Python, a popular high-level programming language.
2. Pandas: The program uses the Pandas library to handle data manipulation and analysis.
3. Geopy: The program uses the Geopy library to calculate geodesic distances between GPS coordinates.
4. DateTime: The program uses the DateTime library to handle timestamp calculations.
5. Random: The program uses the Random library to generate random GPS coordinates and timestamps.
6. Timedelta: The program uses the Timedelta library to calculate time differences.

Additionally, the program uses various Python features, such as:

1. Functions: The program defines several functions to organize and reuse code.
2. Loops: The program uses for loops to iterate over data and perform calculations.
3. Conditional statements: The program uses if statements to handle different scenarios and edge cases.
4. Data structures: The program uses lists and DataFrames to store and manipulate data.

Overall, the program leverages a combination of Python libraries and features to simulate GPS data generation, distance

TEAM MEMBER

BHAVANI V R
USN:1AT22EC013

MENTOR:
SHILPA M N

ATRIA INSTITUTE OF TECHNOLOGY
BENGALURU

THANK YOU