Karnataka ReddyJana Sangha®

VEMANA INSTITUTE OF TECHNOLOGY

Approved by AICTE - New Delhi, Affiliated to VTU Belagavi & Recognized by Govt. Of Karnataka
#1, Mahayogi Vemana Road, 3rd Block, Koramangala, Bengaluru - 560034.
www.vemanait.edu.in

Accredited by NBA

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

# Project

# On

# GPS Toll based system simulation using python

|  |  |
|---|---|
| **NAME OF THE STUDENT** | **: BHARATI** |
| **USN NO** | **: 1VI22EC025** |
| **COURSE/ SEC** | **: ECE/4th SEM** |
| **Email id** | **: bharati.ec2022@vemanait.edu.in** |

## Mentors Details

Internal Mentor : Dr.Girish N

Email id : girish.n@vemanait.edu.in

External Mentor : Sriharsha Gajavalli

Email id : sriharsha.gajavalli@unnatiindustrialtraining2024.com.

## Objective

- Simulate toll calculation using GPS data.

- Account for dynamic factors: traffic congestion, vehicle type, distance travelled. - Visualize toll zones, vehicle routes, and traffic conditions on an interactive map.

## Abstract

The project is a dynamic toll pricing and management system implemented using Python. It calculates toll charges based on several factors such as toll zone, traffic congestion, vehicle type, and distance travelled. It also calculates penalties for speeding and toll waivers based on vehicle status at toll zones.

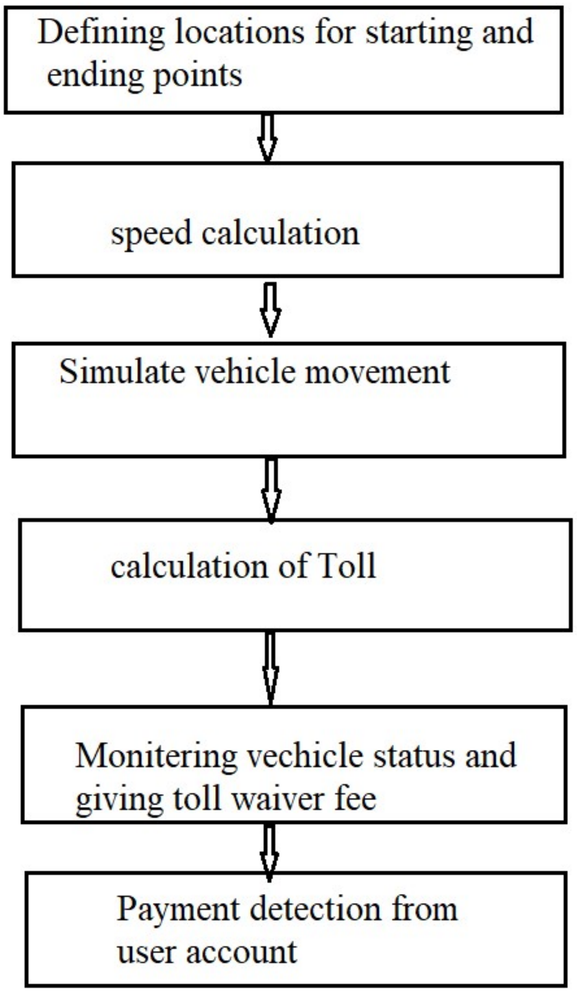
Dynamic Pricing and Congestion Management: The toll rates are adjusted dynamically based on congestion levels, time slots, and vehicle types. A peak hour multiplier increases the toll fee during high traffic periods, while frequent users benefit from discounts if their toll fees exceed a specified threshold.

5. Real-time Vehicle Tracking and Speed Monitoring: The system tracks the number of vehicles on the toll road at any given time and monitors their speed, ensuring adherence to speed limits set for different road sections.

6. Emergency Contingencies: The system includes mechanisms to handle emergencies, such as stationary vehicles. Alerts are generated if a vehicle remains stationary for an extended period.

7. User Interface and Interaction: The system provides a Command-Line Interface (CLI) for user interaction. Users can add vehicles, simulate toll passages, view transaction history, query the number of vehicles on the toll road, check speed limits, and view the current location on a map

**Flow of the project**


Defining locations for starting and ending points
speed calculation
Simulate vehicle movement
calculation of Toll
Monitering vechicle status and giving toll waiver fee
Payment detection from user account

---

**Detailed Explanation**

**Defining locations for starting and ending points** : The variable "locations_coords" is a dictionary that maps city names ("Meerut", "Mathura", "Gurugram", "Agra") to their respective coordinates as tuples of latitude and longitude.

**Defining Toll Zones** :The variable “toll_zones” is a dictionary that defines four toll zones (“Toll Zone 1”, “Toll Zone 2”, “Toll Zone 3”, “Toll Zone 4”) using “shapely.geometry.Polygon” objects with coordinates specifying the vertices of each polygon.

**Calculation of distance between two coordinates and stimulate vehicle movement** :The“calculate_distance” function calculates the distance between two geographical coordinates using the Haversine formula. The coordinates are given as tuples of latitude and longitude, and the function returns the distance in kilometres.

**Formula and Calculation The dynamic pricing factor for each vehicle type is calculated using the following formula:**

**Price per km = Congestion Level × Base Rate (eqn. 1)**

where the base rates are defined as follows:

Car: 0.25 INR/km

Truck: 0.50 INR/km

SUV: 0.35 INR/km

Ambulance: 0.00 INR/km

For example, if the congestion level at the start location is 0.4 (40% congestion), the dynamic pricing factors for each vehicle type would be:

Car: 0.4 × 0.25 = 0.10 INR/km

Truck: 0.4 × 0.50 = 0.20 INR/km

SUV: 0.4 × 0.35 = 0.14 INR/km

Ambulance: 0.4 × 0.00 = 0.00 INR/km 6

Toll Calculation The toll for each toll zone passed is calculated by multiplying the distance travelled within the zone by the dynamic pricing factor, adjusted by a zonespecific multiplier:

**Dynamic Amount = Price per km × Distance × Zone Multiplier (eqn. 2)**

Zone multipliers are predefined as follows: Toll Zone 1: 1.55 Toll Zone 2: 1.25 Toll Zone 3: 1.35 Toll Zone 4: 1.45 For example, if a Car travels 5 km within Toll Zone 1, the dynamic amount would be:

Dynamic Amount = 0.10 × 5 × 1.55 = 0.775 INR

Monitoring Speed Limits and Imposing Penalties The system monitors vehicle speeds in different sections of the route and imposes penalties if the vehicle exceeds the speed limit. Speed limits are predefined for different vehicle types and sections.

**Technologies Used:**

- **Python**: Utilize Python for the core logic, data management, and user interface (if text-based).
- **Folium**: Generate interactive maps to visualize vehicle locations and routes.
- **Haversine Formula**: Calculate distances between GPS coordinates to determine toll fees accurately.
- **Database (optional)**: Use SQLite or other database systems to store vehicle information and transaction history persistently.

**Benefits of Simulation:**

• **Educational Purposes**: Learn about GPS technology, distance calculation algorithms, and transaction management.

• **Prototyping:** Test and refine ideas for a real-world GPS-based toll system before implementation.

• **Demonstration**: Showcase concepts and functionalities to stakeholders or potential users.

**Libraries Used**

1**. requests**: Used to fetch IP information to determine approximate location coordinates.

2. **folium:** Generates interactive maps for visualizing location data.

3. **datetime, time**: Handles date and time operations.

4**. math**: Provides mathematical functions like math.radians() for converting angles from degrees to radians, and trigonometric functions for calculations.

5**. simpy**: Implements discrete-event simulation, useful for modeling vehicle movements over time.

6. **random**: Although imported, it's not extensively used in the provided script.

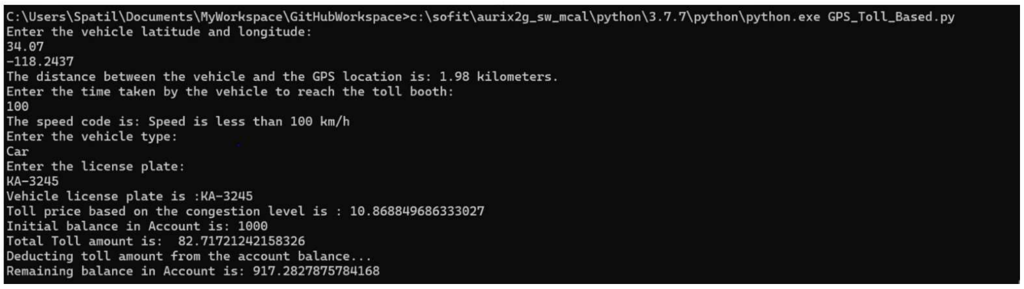
7. **pandas, geopandas**: Handles data structures and spatial data operations.

9. **geopy.distance.geodesic**: Calculates geographical distances between coordinates.

10.**selenium:** Automates web browser interaction for displaying generated maps.
11.**webdriver_manager.chrome**: Manages the Chrome Driver for Selenium.

12.matplotlib.pyplot: Provides plotting functionalities, though not utilized in this script.

## Glimpse of the output



```
C:\Users\Spatil\Documents\MyWorkspace\GitHubWorkspace>c:\sofit\aurix2g_sw_mcal\python\3.7.7\python\python.exe GPS_Toll_Based.py
Enter the vehicle latitude and longitude:
34.07
-118.2437
The distance between the vehicle and the GPS location is: 1.98 kilometers.
Enter the time taken by the vehicle to reach the toll booth:
100
The speed code is: Speed is less than 100 km/h
Enter the vehicle type:
Car
Enter the license plate:
KA-3245
Vehicle license plate is :KA-3245
Toll price based on the congestion level is : 10.868849686333027
Initial balance in Account is: 1000
Total Toll amount is:  82.71721242158326
Deducting toll amount from the account balance...
Remaining balance in Account is: 917.2827875784168
```

## Conclusion

The GPS Toll-Based System Simulation project demonstrates the feasibility of using GPS technology for toll management. Using Python, we created a system that tracks vehicle locations, calculates tolls based on zones, and manages the tolling process. The simulation accurately tracks vehicles, calculates tolls dynamically, and logs transactions in real-time. It includes features for speed monitoring and a user-friendly command-line interface. Future work includes integrating real GPS data, optimizing for scalability, and enhancing security and privacy. Developing a graphical interface or mobile app could improve user experience. The project serves as a proof of concept for a modern toll collection system. It shows the potential for technology to streamline toll collection and improve traffic flow. Overall, the project lays a solid foundation for future enhancements and real-world applications. This successful simulation highlights the benefits of combining GPS and Python for efficient road management.