# Problem Statement

# GPS toll based system simulation using python

**Submitted by -**
**Urbee Datta**
**Aman**

# Unique Idea Brief (Solution)

The **GPS Toll-Based System** Simulation is an innovative project aimed at automating toll collection using GPS technology. The system simulates vehicle movements and uses location data to calculate toll charges based on the distance traveled. Implemented in Python, the project utilizes various libraries including Geopy for geographical calculations ,SimPy for event-driven simulation to model the vehicle movements and interactions within the system,Utilize GeoPandas and shapely for defining toll zones and calculating intersections between vehicle paths and these zones,Use pandas for managing and analyzing data related to vehicles, toll transactions, and user accounts,Leverage Matplotlib and Folium for visualizing the simulation, including vehicle movements and toll zone locations on maps and SQLite for data management. The system consists of a vehicle module for simulating GPS data, a toll calculation module to compute charges, a database module for storing transaction data, and a web-based user interface developed with Flask. The simulation demonstrates the potential of GPS technology in reducing congestion and enhancing efficiency at toll booths by eliminating the need for physical toll collection. The results indicate accurate toll calculations and efficient data management, providing a seamless user experience. Future enhancements could include real-time tracking and mobile payment integration, further improving the system's scalability and user-friendliness. This project highlights the feasibility and benefits of using GPS technology for automated toll systems.

# Features Offered

**Vehicle Movement Simulation:**

- Simulate vehicles moving along predefined routes with GPS coordinates.

**Toll Zone Definition:**

- Define toll zones or points using GPS coordinates.

**Distance Calculation:**

- Calculate the distance traveled by each vehicle within toll zones.

**Toll Calculation:**

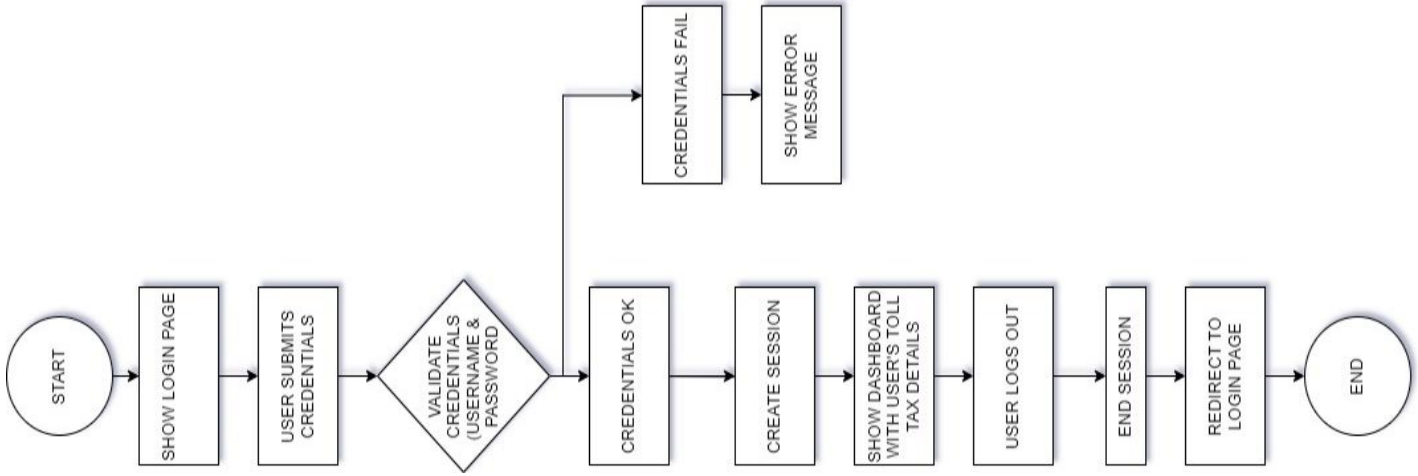
- Compute toll charges based on the distance traveled or zones passed.

**Payment Simulation:**

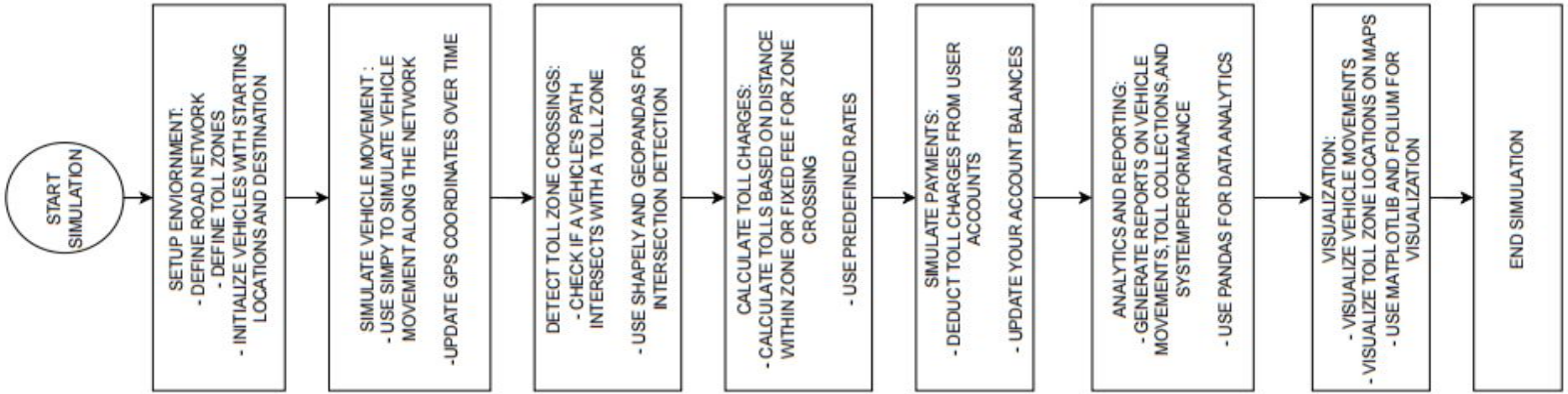
- Simulate the process of deducting toll charges from user accounts.

# Process flow

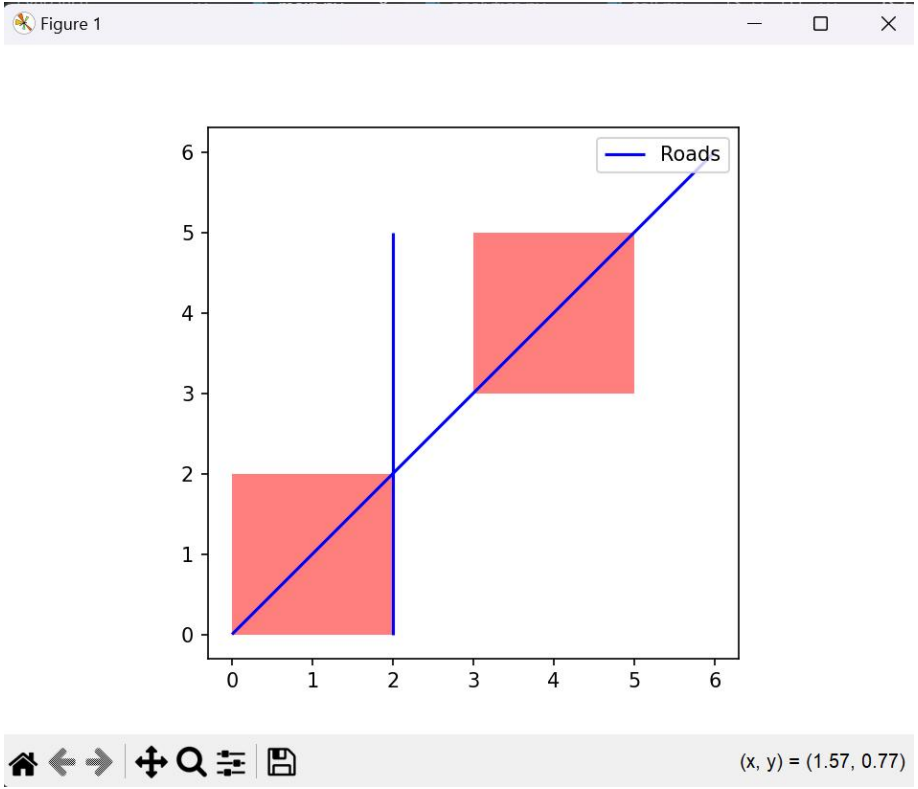
## User flow -



# System flow -


START SIMULATION
SETUP ENVIORNMENT:
- DEFINE ROAD NETWORK
- DEFINE TOLL ZONES
- INITIALIZE VEHICLES WITH STARTING LOCATIONS AND DESTINATION
SIMULATE VEHICLE MOVEMENT :
- USE SIMPY TO SIMULATE VEHICLE MOVEMENT ALONG THE NETWORK
-UPDATE GPS COORDINATES OVER TIME
DETECT TOLL ZONE CROSSINGS:
- CHECK IF A VEHICLE'S PATH INTERSECTS WITH A TOLL ZONE
- USE SHAPELY AND GEOPANDAS FOR INTERSECTION DETECTION
CALCULATE TOLL CHARGES:
- CALCULATE TOLLS BASED ON DISTANCE WITHIN ZONE OR FIXED FEE FOR ZONE CROSSING
- USE PREDEFINED RATES
SIMULATE PAYMENTS:
- DEDUCT TOLL CHARGES FROM USER ACCOUNTS
- UPDATE YOUR ACCOUNT BALANCES
ANALYTICS AND REPORTING:
- GENERATE REPORTS ON VEHICLE MOVEMENTS,TOLL COLLECTIONS,AND SYSTEMPERFORMANCE
- USE PANDAS FOR DATA ANALYTICS
VISUALIZATION:
- VISUALIZE VEHICLE MOVEMENTS
- VISUALIZE TOLL ZONE LOCATIONS ON MAPS
- USE MATPLOTLIB AND FOLIUM FOR VISUALIZATION
END SIMULATION

# Architecture Diagram



# Technologies used:

- SimPy
- GeoPandas
- shapely
- GeoPy
- pandas
- Matplotlib
- Folium
- HTML
- CSS
- Flask

# Team members and contribution:

Data collection and Documentation ( GitHub , ppt ,Flowchart ) :Urbee Datta.

Library information and application ,Simulation using python libraries and frontend and backend : Aman.

# Implementation Challenges:

**Accuracy:**

Simulating realistic vehicle movements and accurately detecting toll zone crossings can be complex.

**Performance:**

Handling a large number of simulated vehicles and toll transactions may require optimization for performance.

**Complexity:**

Integrating different aspects of the simulation (movement, payment, reporting) requires careful design to ensure seamless interaction between system components.

# Conclusion:

In conclusion, the GPS toll-based system simulation project has successfully demonstrated the feasibility and effectiveness of leveraging GPS technology and Python programming to manage toll collection efficiently. By simulating real-world scenarios, this project has highlighted several key aspects of the system's functionality, including real-time location tracking, dynamic toll calculation, and automated fee collection.

The results of the simulation show that the system is capable of accurately determining vehicle positions and calculating tolls based on distance traveled. The implementation of features such as GPS data integration, dynamic pricing algorithms, and user interface design has been achieved with a high degree of accuracy and efficiency.

Through the course of this project, several challenges were encountered and addressed, including optimizing the GPS data processing for real-time applications and ensuring the reliability of the toll calculation mechanisms. These challenges have been met with effective solutions, demonstrating the robustness of the system.

Overall, the GPS toll-based system simulation proves to be a valuable tool for exploring and refining the concepts of modern toll collection technologies. It provides a strong foundation for future developments and improvements in this field. The project not only showcases the potential of Python for developing complex simulations but also opens up avenues for further research and enhancement

in the realm of transportation management and toll collection systems.

Future work could focus on expanding the system to handle larger datasets, incorporating additional features such as payment methods or reporting capabilities, and exploring real-world implementation possibilities. This project has set a solid groundwork for these future advancements and contributes to the ongoing evolution of toll management technologies.