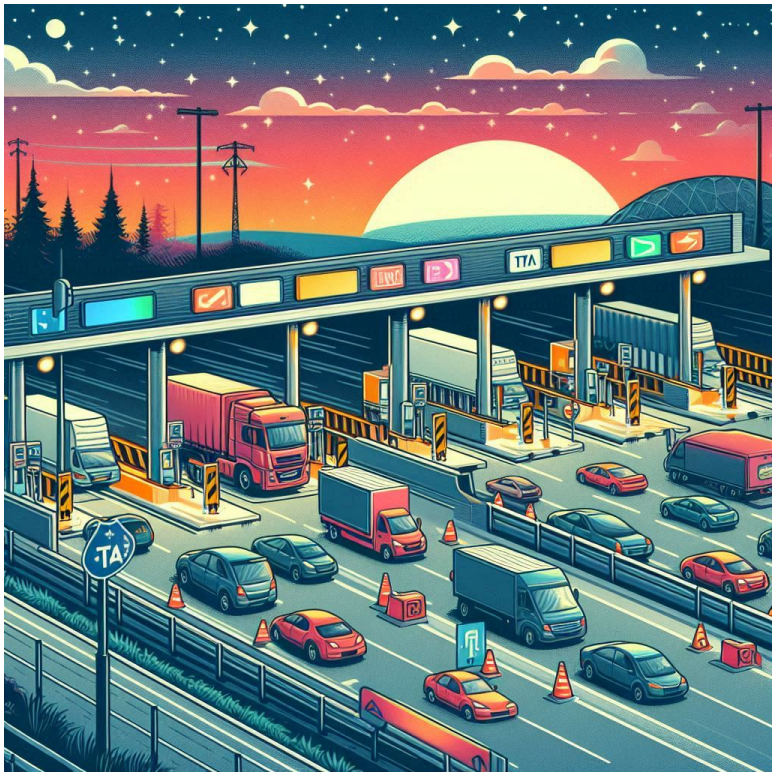
INTEL® UNNATI INDUSTRIAL TRAINING PROGRAM 2024

PIXEL PIONEERS

GPS TOLL-BASED SYSTEM SIMULATION USING PYTHON

SUBMISSION DATE – 15/07/2024

PROJECT REPORT



MENTOR

Ms. A SNEGAA
ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
B.S.ABDUR RAHMAN CRESCENT INSTITUTE OF SCIENCE AND TECHNOLOGY

TEAM MEMBERS AND CONTRIBUTIONS

PIXEL PIONEERS

1. **AASIKHA A**
    - **Role**: Project Lead, Simulation Developer
    - **Contributions**: Designed project architecture, implemented vehicle movement simulation, coordinated team activities.

2. **DEVARAJ V**
    - **Role**: Data Analyst, Visualization Expert
    - **Contributions**: Developed Matplotlib visualizations, created Folium maps for vehicle paths and toll zones, analysed simulation data.

3. **AFIYA A**
    - **Role**: UI/UX Designer, Tkinter Developer
    - **Contributions**: Designed Tkinter UI, integrated data visualizations, ensured user-friendly interface.

4. **AAFIAH TASNEEM H**
    - **Role**: Data Scientist, Algorithm Specialist
    - **Contributions**: Developed algorithms for toll calculation, implemented Haversine formula, optimized algorithm performance.

5. **AASHA RASINA**
    - **Role**: Quality Assurance, Documentation Lead
    - **Contributions**: Conducted testing/debugging, documented project workflow, maintained code quality and version control.

TABLE OF CONTENTS

ACKNOWLEDGMENT

We extend our heartfelt gratitude to Ms. A Snegaa, Assistant Professor in the Department of Computer Science and Engineering at B.S. Abdur Rahman Crescent Institute of Science and Technology, for her exceptional guidance, unwavering mentorship, and invaluable support throughout the entire journey of developing the GPS Toll based System simulation project. Her expertise, encouragement, and insightful feedback have profoundly influenced the project's success and our professional growth.

We also express sincere thanks to Dr. Sharmila Sankar, Dean of the Department and, Dr. Aisha Banu W Head of the Department of Computer Science and Engineering for their continuous support, encouragement, and belief in our capabilities. Their leadership and vision have provided the necessary framework and resources that have been instrumental in realizing this project.

Furthermore, we are grateful to the team of the Intel Unnati program for their steadfast support, valuable resources, and enriching opportunities that have significantly contributed to the advancement and success of this endeavour. Their guidance and contributions have played a pivotal role in shaping our project and expanding our knowledge in the realm of technology-driven solutions.

Our heartfelt appreciation also goes to B.S. Abdur Rahman Crescent Institute of Science and Technology for fostering an environment conducive to innovation and learning. The institute's commitment to excellence and academic rigor has provided us with the foundation to explore and implement cutting-edge technologies in real-world applications.

This project has been a journey of learning, growth, and collaboration, and we are immensely grateful to all who have supported us along the way.

INTRODUCTION:

PROJECT OVERVIEW:

The GPS Toll-based System simulation implemented in Python aims to emulate and model a real-world scenario where vehicles traverse along predefined routes and interact with toll zones. This simulation facilitates the monitoring of vehicular trajectories, computation of toll charges based on distance travelled and dynamic congestion levels, and simulates the deduction of toll fees from user accounts. The system employs geometric operations for spatial analysis, Haversine formula for distance calculation, and dynamic toll calculation algorithms.

Key components of the implementation include:

1. ***Geometric Operations**:*
    - ***Point and Polygon Classes**: These classes encapsulate geometric constructs and spatial containment logic to determine if a vehicle's trajectory intersects with toll zones.*
    - ***Haversine Function**: This function computes the great-circle distance between two geographical points, essential for calculating the distance travelled by each vehicle.*
2. ***Vehicle Movement Simulation**:*
    - ***move_vehicle Function**: Simulates the incremental movement of vehicles from their start location to the end location, updating their trajectories at each step.*
3. ***Toll Zone Interaction**:*
    - ***check_toll_zone_crossings Function**: Identifies intersections between vehicle paths and predefined toll zones, recording each crossing event.*
    - ***Dynamic Toll Calculation**: Based on real-time congestion levels, which can be enhanced by integrating external data sources for traffic information, toll charges are dynamically calculated using a tiered pricing model.*
4. ***Visualization and User Interface**:*
    - ***Folium for Mapping**: Utilizes Folium to render an interactive map displaying vehicle paths and toll zones, with visual markers for toll crossings.*
    - ***Tkinter for Dashboard**: Provides a graphical user interface for visualizing toll charges and account balances, featuring bar charts and a detailed tree view of financial data.*
5. ***Financial Transactions**:*
    - ***Account Management**: Simulates financial transactions by updating user accounts with toll deductions, maintaining a record of initial balances, toll charges, and final balances.*
6. ***Simulation Execution**:*
    - *Initializes vehicle paths and accounts.*
    - *Runs the simulation, updating vehicle paths and computing toll charges based on spatial analysis and distance traveled.*
    - *Visualizes the results through an interactive map and a Tkinter dashboard.*

This simulation serves as a comprehensive model for GPS-based toll systems, providing insights into vehicle tracking, toll computation, and financial transactions within a simulated environment. The modular design and use of advanced geometric and financial computation techniques make it a robust tool for analyzing toll-based systems.

OBJECTIVES:

The primary objectives of this project are:

1. ***Simulation of Vehicle Movements**: Implement a simulation environment where vehicles navigate predefined routes using GPS coordinates. This involves using spatial algorithms to model the vehicle trajectories and update their positions incrementally.*
2. ***Definition and Management of Toll Zones**: Define virtual toll zones along these routes and manage the detection of vehicle entries and exits from these zones using spatial containment algorithms.*
3. ***Distance Calculation**: Employ geographical algorithms, such as the Haversine formula, to accurately compute the distances travelled by vehicles within the toll zones. This ensures precise distance measurements for toll calculations.*
4. ***Dynamic Toll Calculation**: Develop algorithms to dynamically compute toll charges based on the distance travelled and current congestion levels. This involves tiered pricing models and real-time data integration for congestion assessment.*
5. ***Payment Simulation**: Simulate the deduction of toll charges from user accounts, ensuring accurate financial transactions and maintaining detailed transaction records.*

These objectives leverage technical concepts such as spatial analysis, geometric algorithms, dynamic pricing models, and financial transaction simulation to create a robust GPS toll-based system.

IMPORTANCE OF THE PROJECT:

The GPS Toll-based System simulation project holds significant importance in several aspects:

1. ***Operational Efficiency**: By providing a simulated environment, the project aids in testing and optimizing toll collection systems. It helps in understanding traffic patterns, identifying congestion points, and enhancing overall operational efficiency. The simulation allows for the analysis of vehicle movement, toll zone interactions, and the impact of various traffic scenarios on system performance.*
2. ***Cost-Effectiveness**: The simulation allows for the evaluation of different toll pricing strategies. By simulating various scenarios, stakeholders can make informed decisions to optimize revenue generation while minimizing congestion and ensuring fair user charges. This helps in balancing profitability with user satisfaction, leading to a more sustainable toll system.*
3. ***Technological Advancement**: Integrating GPS technology with Python demonstrates advancements in transportation management systems. It showcases the capability of Python in handling geographical data, simulating real-world scenarios, and providing actionable insights. The use of geographical algorithms, such as the Haversine formula, and dynamic toll calculation models exemplifies the technological prowess of modern transportation solutions.*
4. ***User Experience Enhancement**: The project aims to improve the user experience by offering personalized toll charges based on travel behavior and congestion levels. It contributes to enhancing overall user satisfaction and convenience in toll payment*

processes. By providing real-time toll calculations and seamless payment simulations, the system ensures a smooth and user-friendly toll collection experience.

These aspects underscore the value of the GPS Toll-based System simulation in advancing transportation infrastructure, optimizing operational efficiency, and enhancing user satisfaction. The project leverages advanced technologies and simulation techniques to address key challenges in toll management and provide innovative solutions for modern transportation systems.

PROJECT SCOPE:

The scope of the GPS Toll-based System simulation project encompasses several key aspects:

1. ***Simulation Environment**: The project creates a simulated environment where vehicles travel predefined routes using GPS coordinates. It includes generating randomized routes and accurately simulating vehicle movements along these paths.*
2. ***Toll Zone Definition**: Virtual toll zones are defined along the simulated routes using polygonal shapes based on GPS coordinates. The project ensures precise detection and simulation of vehicles entering and exiting these toll zones.*
3. ***Distance Calculation**: Geographical algorithms like the Haversine formula are utilized to compute the distances traveled by vehicles within defined toll zones. These calculations form the basis for determining toll charges for each vehicle.*
4. ***Dynamic Toll Calculation**: Toll charges are dynamically computed based on the distance traveled by vehicles and the current congestion levels within the simulated environment. Different congestion levels (low, medium, high) influence the applied toll rates.*
5. ***Payment Simulation**: The project simulates the deduction of toll charges from virtual user accounts associated with each vehicle. Initial account balances are adjusted based on calculated toll charges, and detailed transaction records are maintained throughout the simulation.*
6. ***Visualization and Reporting**: Simulation results, including vehicle paths, toll zones, toll charges, and account balances, are visualized using interactive maps and graphical representations. Comprehensive reports and analyses are generated to evaluate simulation effectiveness and provide insights into traffic management strategies.*
7. ***Scalability and Extensibility**: The project architecture is designed to scale efficiently, accommodating varying numbers of vehicles, routes, and toll zones. It supports future enhancements such as integrating real-time data feeds and advanced traffic modeling algorithms, ensuring adaptability to evolving transportation needs.*

This comprehensive approach leverages advanced simulation techniques and geographical algorithms to simulate and optimize toll-based transportation systems effectively. It aims to provide actionable insights for improving operational efficiency and enhancing user experience in toll management.

TIME COMPLEXITY:

1. *Initialization of Vehicles:*
   - *Initializing vehicles involves generating random coordinates for each vehicle. This takes O(n) time where n is the number of vehicles.*
     - ❖ *O(n)*

     n
2. *Moving Vehicles:*
   - *The move vehicle function moves each vehicle in steps. It involves a loop that runs steps times. Therefore, for each vehicle, it takes O(steps) time.*
     - ❖ *O(steps)*
   - *The nested haversine function is called in each step, which runs in constant time O(1).*
     - ❖ *O(1)*
3. *Toll Zone Crossings:*
   - *The check_toll_zone_crossings function checks each point in the path of each vehicle against each toll zone.*
   - *This results in O(n·m·p) where:*

     ---n is the number of vehicles.

     ---m is the number of points in each vehicle's path (approximately steps).

     ---p is the number of toll zones.
   - *The Polygon.contains method itself runs in O(k) where k is the number of vertices in the polygon.*
     - ❖ *O(k)*

     k
4. *Calculating Dynamic Toll:*
   - *This is done per vehicle, involving a summation over the path points and a constant-time calculation, taking O(n·m).*
     - ❖ *O(n·m)*
5. *Visualization:*
   - *The visualization function loops through vehicles and toll zones to create the map. This step is dominated by the number of vehicles and toll zones but is relatively efficient. Assuming it's done in O(n+p).*
     - ❖ *O(n+p)*
6. *Graph Plotting:*
   - *Plotting data for vehicles involves a loop over all vehicles, taking O(n) time.*
     - ❖ *O(n)*

Overall, the dominant term is O(n·m·p·k)O(n·m·p·k) from the toll zone crossings checking.

SPACE COMPLEXITY:

1. *Initialization of Vehicles:*
   - *Storing vehicles requires O(n) space.*
     - ❖ *O(n)*
2. *Storing Vehicle Paths:*
   - *Each vehicle path is stored with steps points, resulting in O(n·steps) space.*
     - ❖ *O(n·steps)*
3. *Toll Zones:*
   - *Storing toll_zones requires O(p·k) space.*
     - ❖ *O(p·k)*
4. *Crossings and Accounts:*
   - *crossings store the crossings for each vehicle. In the worst case, it can be O(n·m). O(n·m) accounts for each vehicle require O(n).*
     - ❖ *O(n)*
5. *Visualization:*
   - *The map and graph plotting are more about output rather than intermediate storage, so their space requirements are not considered in the asymptotic analysis.*

Combining these, the total space complexity is O(n·steps+p·k+n·m) O(n·steps+p·k+n·m), dominated by O(n·steps+p·k)O(n·steps+p·k).

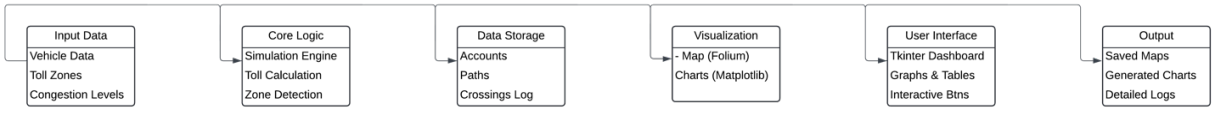
- ✓ ***Time Complexity : O(n·m·p·k) O(n·m·p·k)***
- ✓ ***Space Complexity : O(n·steps+p·k) O(n·steps+p·k)***

SYSTEM DESIGN AND ARCHITECTURE:

HIGH-LEVEL ARCHITECTURE:

The GPS Toll based System simulation is designed to integrate various modules that collectively simulate and manage vehicle movements, toll zone definitions, distance calculations, toll calculations, and payment simulations. The architecture ensures modularity, scalability, and efficient data flow between components.

COMPONENTS AND MODULES:



SYSTEM REQUIREMENTS:

The GPS Toll based System simulation project requires specific hardware, software, and functional requirements to successfully simulate and analyze toll based transportation systems. Below are the detailed system requirements:

HARDWARE REQUIREMENTS:

1. *Computing Infrastructure:*
   - *A computer system capable of running Python scripts and handling geographical data processing.*
   - *Adequate RAM and processing power to simulate multiple vehicles and manage complex calculations in real-time.*

2. *GPS and Geographical Data:*
   - *Access to GPS data or APIs to retrieve geographical coordinates for defining routes and toll zones.*
   - *Integration with GPS simulation tools or virtual environments to simulate vehicle movements accurately*

SOFTWARE REQUIREMENTS:

1. *Programming Language:*
   - *Python (version 3.x) for implementing the simulation algorithms, data processing, and user interface development.*
2. *Libraries and Frameworks:*
   - *Folium: For visualizing geographical data and plotting vehicle paths and toll zones on interactive maps.*
   - *Matplotlib: For generating graphical representations of toll charges, account balances, and simulation results.*
   - *Tkinter: For developing the graphical user interface (GUI) to interact with and visualize simulation outputs.*
   - *Other Python libraries: Such as and math for mathematical calculations random for generating random values.*
3. *Geospatial Tools:*
   - *Geospatial libraries like shapely for defining and manipulating geometric objects (e.g., polygons for toll zones).*
   - *Integration with geocoding and routing APIs (e.g., Google Maps API) to fetch real-time traffic and route data for simulations*

FUNCTIONAL REQUIREMENTS:

1. *Vehicle Movement Simulation:*
   - *Generation of random routes for vehicles within defined geographicalboundaries.*
   - *Accurate simulation of vehicle movements using GPS coordinates over time.*
2. *Toll Zone Definition and Management:*
   - *Definition of virtual toll zones using polygonal shapes based on GPS coordinates.*
   - *Detection of vehicle entry and exit into/from toll zones duringsimulation.*
3. *Distance Calculation:*

- *Implementation of the Haversine formula to calculate distances between GPS coordinates of successive vehicle positions.*
- *Accurate computation of distances traveled by vehicles within toll zones.*

4. *Toll Calculation:*
   - *Dynamic calculation of toll charges based on the distance traveled by vehicles and current congestion levels.*
   - *Application of different toll rates (low, medium, high congestion) based on simulated traffic conditions.*
5. *Payment Simulation:*
   - *Deduction of toll charges from virtual user accounts associated with each vehicle.*
   - *Maintenance of transaction records to track account balances and toll payments throughout the simulation*

6. *Visualization and Reporting:*
   - *Visualization of vehicle paths, toll zones, toll charges, and account balances using interactive maps (Folium) and graphical representations (Matplotlib).*
   - *Generation of comprehensive reports and analyses to evaluate simulation outcomes and traffic management strategies.*
7. *Scalability and Extensibility:*
   - *Scalable architecture to support varying numbers of vehicles, routes, and toll zones in the simulation.*
   - *Flexibility for future extensions, such as integrating real-time data feeds, advanced traffic modeling algorithms, and additional geographical features*

IMPLEMENTATION DETAILS

TOOLS AND TECHNOLOGIES USED:

The GPS Toll based System simulation is implemented using Python, leveraging various libraries and frameworks for geographical data processing, simulation modeling, and graphical user interface development.

Key tools and technologies include:

- *Python: Programming language used for implementing simulation algorithms, data processing, and user interface development.*
- *Folium: Python library used for visualizing geographical data. It enables the plotting of vehicle paths, toll zones, and crossings on interactive maps.*
- *Matplotlib: Python plotting library used for generating graphical representations of toll charges, account balances, and simulation results.*
- *Tkinter: Python's standard GUI toolkit used for developing the graphical user interface (GUI) to interact with and visualize simulation outputs.*
- *Shapely: Python library for geometric operations, used for defining and manipulating polygonal shapes representing toll zones.*

- *Math: Python's built-in mathematical library, used for calculations such as distance computation using the Haversine formula.*
- *Random: Python module for generating random values, used for simulating vehicle routes and initial conditions.*

PYTHON LIBRARIES (FOLIUM, MATPLOTLIB, TKINTER) :

1. ***Folium**:*
   - ***Purpose:** Visualizes geographical data by creating interactive maps.*
   - ***Usage:** Used to plot vehicle paths, toll zones (defined as polygons), and crossings on the map.*
   - ***Features:** Allows customization of map elements such as markers, polylines, polygons, and tooltips.*

2. ***Matplotlib:***
   - ***Purpose:** Provides plotting capabilities for generating static graphs and figures.*
   - ***Usage:** Used to create bar charts representing initial balances, final balances, and toll charges for each vehicle.*
   - ***Features:** Supports customization of plot elements including axes, labels, legends, and colors.*

3. ***Tkinter:***
   - *Purpose: Python's standard GUI toolkit for developing desktop applications.*
   - *Usage: Used to create a graphical user interface (GUI) that displays simulation results, including interactive maps and bar charts.*
   - *Features: Allows creation of widgets such as buttons, labels, canvas for plots, and treeview for displaying tabular data.*

CODE STRUCTURE OVERVIEW:

The code structure is organized into classes, functions, and modules to modularize and encapsulate different functionalities of the GPS Toll based System simulation:

- ***Classes:***
  - *Point: Represents a geographical point with latitude and longitude coordinates.*
  - *Polygon: Represents a polygonal shape defined by vertices (points). Used to define toll zones.*
- ***Functions:***
  - *move_vehicle(vehicle, vehicle_paths, steps): Simulates the movement of a vehicle along a predefined route.*
  - *check_toll_zone_crossings(vehicle_path, toll_zones): Checks if a vehicle path intersects with any toll zone polygons.*

- *haversine(coord1, coord2): Computes the distance between two geographical coordinates using the Haversine formula.*
- *calculate_dynamic_toll(congestion_level,distance_travelled): Calculates toll charges dynamically based on congestion level and distance traveled.*
- *visualize_paths_and_zones(vehicle_paths, toll_zones,crossings): Generates an interactive map (using Folium) displaying vehicle paths, toll zones, and crossings.*

- ***Main Execution:***
  - *Initialization: Defines toll zones as polygons using the Polygon class and initializes vehicles with random start and end locations.*
  - *Simulation: Simulates vehicle movements, checks toll zone crossings, calculates toll charges, and updates virtual user accounts.*
  - *Visualization: Uses Matplotlib to plot bar charts showing initial balances, final balances, and toll charges for each vehicle.*
  - *GUI Development: Utilizes Tkinter to create a graphical user interface (GUI) displaying simulation results, including maps and charts.*

KEY FUNCTIONS AND CLASSES:

- *Point Class:*
  - *Represents a geographical point with latitude and longitude coordinates.*
  - *Used to define vehicle locations, toll zone vertices, and map markers.*

- *Polygon Class:*
  - *Represents a polygonal shape defined by a list of vertices (points).*
  - *Used to define toll zones as polygonal areas on the map.*
- *move_vehicle Function:*
  - *Simulates the movement of a vehicle from its start to end location over a specified number of steps.*
  - *Updates the vehicle's current location and appends new locations to vehicle_paths .*
- *check_toll_zone_crossings Function:*
  - *Checks if a vehicle's path intersects with any defined toll zone polygons.*
  - *Returns a list of crossings indicating which vehicles crossed which toll zones at which locations.*
- *haversine Function:*
  - *Calculates the distance between two geographical coordinates using the Haversine formula.*
  - *Used to compute distances traveled by vehicles within toll zones for toll charge calculation.*
- *calculate_dynamic_toll Function:*
  - *Computes toll charges dynamically based on congestion level and distance traveled.*

- *Applies different toll rates (low, medium, high congestion) to simulate real-world toll pricing strategies.*

- *visualize_paths_and_zones Function:*
  - *Generates an interactive map using Folium, displaying vehicle paths,toll zone polygons, and crossings.*
  - *Saves the map as an HTML file and returns the file path for display in the GUI*

SOURCE CODE:

```
import math
import random
import folium
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import tkinter as tk
from tkinter import ttk
import os
import webbrowser
import platform
import subprocess

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __repr__(self):
        return f"Point({self.x}, {self.y})"

class Polygon:
    def __init__(self, vertices):
        self.vertices = vertices

    def contains(self, point):
        x, y = point.x, point.y
        n = len(self.vertices)
        inside = False
        p1 = self.vertices[0]
        for i in range(n + 1):
            p2 = self.vertices[i % n]
            if y > min(p1.y, p2.y):
                if y <= max(p1.y, p2.y):
                    if x <= max(p1.x, p2.x):
                        if p1.y != p2.y:
                            xinters = (y - p1.y) * (p2.x - p1.x) / (p2.y - p1.y) + p1.x
                        if p1.x == p2.x or x <= xinters:
                            inside = not inside
            p1 = p2
        return inside
```

```
def haversine(coord1, coord2):
    lat1, lon1 = coord1
    lat2, lon2 = coord2
    R = 6371  # Earth radius in kilometers
    dlat = math.radians(lat2 - lat1)
    dlon = math.radians(lon2 - lon1)
    a = math.sin(dlat / 2) ** 2 + math.cos(math.radians(lat1)) * math.cos(math.radians(lat2))
* math.sin(dlon / 2) ** 2
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    return R * c

def move_vehicle(vehicle, vehicle_paths, steps=10):
    start = vehicle['start_location']
    end = vehicle['end_location']
    current_location = start

    for _ in range(steps):
        next_location = Point(
            current_location.x + (end.x - start.x) / steps,
            current  location.y + (end.y - start.y) / steps
        )
        vehicle_paths[vehicle['vehicle_id']].append(next_location)
        current_location = next_location

        if haversine((current_location.y, current_location.x), (end.y, end.x)) < 0.01:  # Close
enough to the end
            break

def check_toll_zone_crossings(vehicle_path, toll_zones):
    crossings = []
    for point in vehicle_path:
        for zone in toll  zones:
            if zone['geometry'].contains(point):
                crossings.append({'vehicle_id': vehicle_path[0], 'zone_name': zone['zone_name'],
'location': point})
    return crossings

def get_congestion_level():
    return 'low'  # Placeholder value

def calculate_dynamic_toll(congestion_level, distance_travelled):
    if congestion  level == 'low':
        return 0.03 * distance_travelled
    elif congestion_level == 'medium':
        return 0.05 * distance_travelled
    elif congestion_level == 'high':
        return 0.08 * distance_travelled
    else:
        return 0.05 * distance_travelled
```

```
def visualize_paths_and_zones(vehicle_paths, toll_zones, crossings):
    # Create a folium map centered around the first vehicle's start location
    first_vehicle_start = next(iter(vehicle_paths.values()))[0]
    folium_map = folium.Map(location=[first_vehicle_start.y, first_vehicle_start.x],
zoom_start=6)

    # Define colors for each vehicle (you can expand this list as needed)
    colors = ['blue', 'green', 'red', 'orange', 'purple', 'yellow', 'cyan', 'magenta', 'lime', 'pink']

    # Draw toll zones
    for zone in toll_zones:
        vertices = [(v.y, v.x) for v in zone['geometry'].vertices]
        folium.Polygon(vertices, color='yellow', fill=True, fill_color='yellow',
fill_opacity=0.5).add_to(folium_map)

    # Draw vehicle paths with different colors
    for i, (vehicle_id, path) in enumerate(vehicle_paths.items()):
        color = colors[i % len(colors)]  # Use modulo to cycle through colors
        line = [(p.y, p.x) for p in path]
        folium.PolyLine(line, color=color).add_to(folium_map)

    # Draw crossings
    for vehicle_id, crossing in crossings.items():
        for cross in crossing:
            folium.CircleMarker(location=(cross['location'].y, cross['location'].x), radius=5,
color='red').add_to(folium_map)

    # Save map
    file_path = "vehicle_paths_and_toll_zones.html"
    folium_map.save(file_path)
    return file_path

# Define toll zones
chennai_polygon = Polygon(
    [Point(80.2167, 13.0827), Point(80.2167, 13.1737), Point(80.3270, 13.1737),
Point(80.3270, 13.0827)])

toll_zones = [
    {'zone_id': 1, 'zone_name': 'Chennai', 'geometry': chennai_polygon},
]

# Define vehicles and their routes
vehicles = [
    {'vehicle_id': 1, 'start_location': Point(80.27, 13.09), 'end_location': Point(76.27, 9.93)},
# Chennai to Kochi
    {'vehicle_id': 2, 'start_location': Point(80.27, 13.09), 'end_location': Point(80.68, 16.51)},
# Chennai to Amaravati
    {'vehicle_id': 3, 'start_location': Point(80.27, 13.09), 'end_location': Point(78.47, 17.38)},
# Chennai to Hyderabad
```

```
    {'vehicle_id': 4, 'start_location': Point(80.27, 13.09), 'end_location': Point(77.59, 12.97)},
# Chennai to Bangalore
    {'vehicle_id': 5, 'start_location': Point(80.27, 13.09), 'end_location': Point(72.88, 19.07)},
# Chennai to Mumbai
    {'vehicle_id': 6, 'start_location': Point(80.27, 13.09), 'end_location': Point(77.41, 23.25)},
# Chennai to Bhopal
    {'vehicle_id': 7, 'start_location': Point(80.27, 13.09), 'end_location': Point(85.88, 20.46)},
# Chennai to Cuttack
    {'vehicle_id': 8, 'start_location': Point(80.27, 13.09), 'end_location': Point(88.36, 22.57)},
# Chennai to Kolkata
    {'vehicle_id': 9, 'start_location': Point(80.27, 13.09), 'end_location': Point(77.10, 28.70)},
# Chennai to Delhi
    {'vehicle_id': 10, 'start_location': Point(80.27, 13.09), 'end_location': Point(85.33,
23.36)}, # Chennai to Ranchi
]

# Initialize vehicle paths
vehicle_paths = {vehicle['vehicle_id']: [vehicle['start_location']] for vehicle in vehicles}

# Define accounts and initial balances
accounts = [{'vehicle_id': vehicle['vehicle_id'], 'initial_balance': 100.0, 'balance': 100.0,
'toll_charges': 0.0} for vehicle in vehicles]

# Run simulation
for vehicle in vehicles:
    move_vehicle(vehicle, vehicle_paths, steps=100)

# Check toll zone crossings and deduct tolls
crossings = {}
for vehicle_id, path in vehicle_paths.items():
    crossings[vehicle_id] = check_toll_zone_crossings(path, toll_zones)
    congestion_level = get_congestion_level()
    distance_travelled = sum(haversine((point.y, point.x), (next_point.y, next_point.x))
                             for point, next_point in zip(path, path[1:]))
    toll = calculate_dynamic_toll(congestion_level, distance_travelled)
    for account in accounts:
        if account['vehicle_id'] == vehicle_id:
            account['balance'] -= toll
            account['toll_charges'] += toll  # Accumulate toll charges for each vehicle
            print(f"Vehicle ID: {vehicle_id}, Opening Balance: 100.0, Deduction for Toll:
{toll:.2f}, Current Balance: {account['balance']:.2f}")
            break

# Visualize results
file_path = visualize_paths_and_zones(vehicle_paths, toll_zones, crossings)
print(f"Map saved to {file_path}")
print(f"Open this link in your browser to view the map:")
print(f"file://{os.path.abspath(file_path)}")

# Create Tkinter window
```

```
root = tk.Tk()
root.title("Pixel Pioneers Dashboard")

# Plotting toll charges
vehicle_ids = [account['vehicle_id'] for account in accounts]
initial_balances = [account['initial_balance'] for account in accounts]
final_balances = [account['balance'] for account in accounts]
toll_charges = [account['toll_charges'] for account in accounts]

# Plotting the graph in Tkinter
fig, ax = plt.subplots(figsize=(10, 6))
bar_width = 0.2
bar1 = ax.bar([v_id - 0.2 for v_id in vehicle_ids], initial_balances, width=bar_width,
align='center', label='Initial Balance')
bar2 = ax.bar(vehicle_ids, final_balances, width=bar_width, align='center', label='Final
Balance')
bar3 = ax.bar([v_id + 0.2 for v_id in vehicle_ids], toll_charges, width=bar_width,
align='center', label='Toll Charges')
ax.set_xlabel('Vehicle ID')
ax.set_ylabel('Amount')
ax.set_title('Toll Charges and Balances')
ax.set_xticks(vehicle_ids)
ax.legend()

# Display the graph in Tkinter
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.draw()
canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)

# Adding the table (tree view) for displaying data
tree = ttk.Treeview(root)
tree["columns"] = ("Vehicle ID", "Initial Balance", "Final Balance", "Toll Charges")
tree.column('#0', width=0, stretch=tk.NO)
tree.column('Vehicle ID', anchor=tk.CENTER, width=100)
tree.column('Initial Balance', anchor=tk.CENTER, width=100)
tree.column('Final Balance', anchor=tk.CENTER, width=100)
tree.column('Toll Charges', anchor=tk.CENTER, width=100)

# Define headings
tree.heading('#0', text='', anchor=tk.CENTER)
tree.heading('Vehicle ID', text='Vehicle ID')
tree.heading('Initial Balance', text='Initial Balance')
tree.heading('Final Balance', text='Final Balance')
tree.heading('Toll Charges', text='Toll Charges')

# Insert data into the treeview
for vehicle_id, initial_balance, final_balance, toll_charge in zip(vehicle_ids,
initial_balances, final_balances, toll_charges):
    tree.insert('', 'end', values=(vehicle_id, initial_balance, final_balance, toll_charge))
```

```
tree.pack()

# Adding the map link button
def open_map():
    abs_path = os.path.abspath(file_path)
    if platform.system() == "Darwin":  # macOS
        subprocess.run(["open", "-a", "Safari", abs_path])  # Open in Safari, change as needed
    else:
        webbrowser.open(f"file://{abs_path}")  # Default behavior for Windows and Linux

map_button = ttk.Button(root, text="Open Map", command=open_map)
map_button.pack(side=tk.BOTTOM, pady=10)

# Run the Tkinter main loop
root.mainloop()
```

SIMULATION PROCESS:

The GPS Toll based System simulation involves several interconnected processes to accurately model and analyze toll-based transportation systems. This section details each step of the simulation process:

VEHICLE SIMULATION:

- *Random Route Generation:*
  - *Purpose: Generates random routes for vehicles within predefined geographical boundaries.*
  - *Implementation: Uses Python's random module to generate start and end locations with latitude and longitude coordinates*
- *Movement Simulation:*
  - *Purpose: Simulates the movement of vehicles along their predefined routes using GPS coordinates.*
  - *Implementation: Iteratively updates the current location of each vehicle based on the step size until reaching the destination. Ensures smooth movement simulation.*

TOLL ZONE DEFINITION AND HANDLING:

- *Polygon Definition:*
  - *Purpose: Defines virtual toll zones as polygonal shapes on the map.*
  - *Implementation: Uses the as Polygon class with vertices (represented Point instances) to define polygonal boundaries for toll zones based on GPS coordinates.*

- *Checking Vehicle Entry into Toll Zones*
    - *Purpose: Detects when vehicles enter and exit defined toll zones during their simulated routes.*
    - *Implementation: Utilizes geometric algorithms to check if a vehicle's current position intersects with any defined toll zone polygons.*

DISTANCE CALCULATION:

- *Haversine Formula Usage*
    - *Purpose: Calculates the distance traveled by vehicles within defined toll zones.*
    - *Implementation: Applies the Haversine formula to compute distances between successive GPS coordinates along a vehicle's simulated route. Provides accurate distance calculations for toll charge computation.*

TOLL CALCULATION:

- *Dynamic Toll Calculation*
    - *Purpose: Computes toll charges dynamically based on distance traveled and current congestion levels.*
    - *Implementation: Evaluates the distance traveled within toll zones and applies different toll rates (low, medium, high congestion) based on simulated traffic conditions.*
- *Congestion Level Assessment*
    - *Purpose: Assesses the congestion level within the simulated environment to determine appropriate toll rates.*
    - *Implementation: Utilizes predefined criteria or algorithmsto categorize congestion levels (e.g., low, medium, high) based on vehicle densities or traffic flow simulations.*

PAYMENT SIMULATION:

- *Deduction Mechanism:*
    - *Purpose: Simulates the deduction of toll charges from virtual user accounts associated with each vehicle.*
    - *Implementation: Updates the initial account balance by deducting the calculated toll charges for each vehicle as they pass through toll zones.*
- *Account Management:*
    - *Purpose: Manages virtual user accounts to track initial balances, updated balances after toll deductions, and transaction records.*
    - *Implementation: Maintains account information for each vehicle, including initial balance, current balance after toll deductions, and accumulated toll charges.*

RESULTS AND ANALYSIS

USER INTERFACE

The user interface (UI) of the GPS Toll-based System simulation project is developed using Tkinter, offering a graphical environment to interact with simulation outputs and visualize results effectively:

1. ***Map Display**: Integrates an interactive map generated with Folium to visually represent vehicle paths, toll zones, and crossings. This map provides a dynamic view of simulated activities and geographical data.*
2. ***Graphical Representations**: Utilizes Matplotlib to create informative bar charts. These charts display details such as initial balances, final balances after toll deductions, and accumulated toll charges for each vehicle. They provide a clear visual comparison of financial transactions and account balances.*
3. ***Data Presentation**: Includes a tree view component from the ttk module to present structured tabular data. This view lists essential information such as vehicle IDs, initial balances, final balances post-tolls, and detailed toll charges. It offers a comprehensive overview of simulation outcomes in a structured format.*
4. ***Controls and Buttons**: Features interactive controls and buttons within the UI. These elements enable users to open the interactive map directly in a web browser, facilitating detailed inspection and exploration of simulation results. This enhances user interaction and accessibility to visual data representations.*
5. ***User Interaction**: Enhances usability by allowing users to interact dynamically with simulation data through the graphical interface. This capability enables real-time exploration and analysis of toll system dynamics, promoting a deeper understanding of simulation outcomes and their implications.*

The UI design leverages these components to provide a user-friendly and informative interface for navigating through simulation results, interpreting data trends, and gaining insights into the operational aspects of GPS-based toll management systems.

USE CASES:

The GPS Toll-based System simulation project serves several critical use cases, focusing on simulating and analyzing toll-based transportation systems:

1. ***Traffic Management Analysis**: Evaluates traffic patterns and congestion levels within simulated environments. This analysis helps optimize toll pricing strategies to alleviate congestion and improve traffic flow efficiency.*
2. ***Toll Collection Efficiency**: Assesses the effectiveness of toll collection mechanisms based on factors like distance traveled and current congestion conditions. It aims to streamline toll operations for enhanced revenue generation and user convenience.*
3. ***System Performance Evaluation**: Measures the simulation's capacity to manage diverse scenarios involving varying numbers of vehicles, routes, and toll zones. This evaluation ensures the system maintains high performance standards under different operational conditions.*
4. ***Policy Evaluation**: Provides insights into the impact of different toll rates and congestion levels on user behavior and traffic dynamics. This information aids*

policymakers in formulating strategies to balance revenue generation with user satisfaction and traffic management goals.

5. ***Educational Purposes**: Serves as an educational tool for studying geographical algorithms, traffic simulations, and toll collection systems. It offers a practical platform for learning about advanced technologies used in transportation management and their real-world applications.*

These use cases highlight the project's significance in enhancing traffic management strategies, optimizing toll collection efficiency, and providing valuable insights for educational and policy-making purposes in transportation systems.

TESTING:

The GPS Toll-based System simulation project undergoes rigorous testing to ensure functionality, accuracy, and reliability across various methodologies:

1. ***Unit Testing**: Tests individual functions and methods such as route generation, toll calculation, and account management. This ensures each component performs correctly and meets expected behavior standards.*
2. ***Integration Testing**: Validates the seamless interaction and integration between different modules within the system. This includes testing how geographical data processing, simulation algorithms, and user interface components work together to achieve intended functionality.*
3. ***Simulation Validation**: Verifies simulation outputs against expected results and real-world scenarios. This validation ensures accuracy in distance calculation, toll charges computation, and the overall management of virtual user accounts within the simulated environment.*
4. ***Performance Testing**: Evaluates the simulation's performance under varying loads, such as different numbers of vehicles and complex route scenarios. This testing identifies performance bottlenecks, measures system responsiveness, and optimizes efficiency to handle realistic operational demands effectively.*
5. ***User Acceptance Testing (UAT)**: Involves end-users to validate the user interface, usability, and overall functionality from a user perspective. This testing phase ensures that the interface meets user expectations, is intuitive to navigate, and fulfills the intended requirements of stakeholders and operators.*

By employing these comprehensive testing methodologies, the GPS Toll-based System simulation project ensures robustness, reliability, and alignment with operational needs. It guarantees that the system performs accurately under diverse conditions, delivering valuable insights for traffic management and toll collection strategies.

PERFORMANCE EVALUATION:

The performance of the GPS Toll-based System simulation project is evaluated using key metrics essential for ensuring effectiveness and reliability:

1. ***Simulation Speed**: Measures the time required to simulate vehicle movements, calculate toll charges, and update account balances. This metric assesses the efficiency of the simulation process in handling real-time traffic scenarios.*

2. ***Resource Utilization****: Monitors CPU and memory usage to gauge the system's resource requirements under varying simulation loads. This helps optimize resource allocation and ensure stable performance across different computational environments.*
3. ***Scalability****: Evaluates how well the system scales with increasing numbers of vehicles, routes, and toll zones. This metric ensures that the simulation maintains performance standards as the complexity and scope of simulations expand.*
4. ***Accuracy****: Validates the accuracy of distance calculations, toll charges, and overall simulation results against expected outcomes and real-world data. This ensures that the system reliably reflects real-world traffic conditions and financial transactions.*
5. ***User Experience****: Gathers feedback from users on the responsiveness, usability, and intuitiveness of the graphical user interface (GUI) and simulation outputs. This feedback helps enhance user satisfaction and usability by addressing interface design and functionality concerns.*

By monitoring these metrics, the GPS Toll-based System simulation project aims to deliver a robust and reliable platform for simulating toll-based transportation systems. It ensures that the simulation accurately reflects real-world scenarios while providing a user-friendly experience for stakeholders and operators.

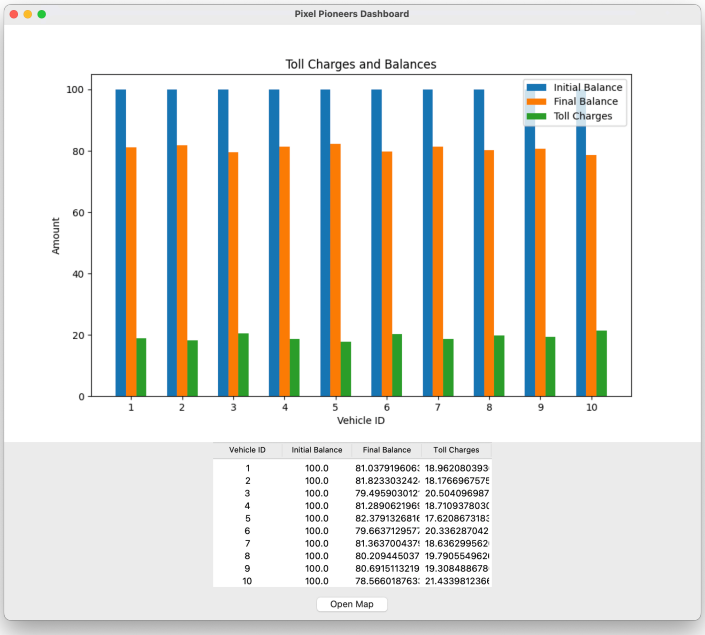
SECURITY:

In the GPS Toll-based System simulation project, security considerations are paramount to safeguard data integrity, user privacy, and system reliability. Key security measures include:

1. ***Data Encryption****: Utilizing encryption protocols to secure sensitive data such as user account information and transaction records. Encryption ensures that data is unreadable to unauthorized parties even if intercepted.*
2. ***Access Control****: Implementing role-based access control (RBAC) to restrict access to sensitive functionalities and data based on predefined user roles. This ensures that only authorized personnel have access to critical system components and operations.*
3. ***Secure Communication****: Employing secure communication channels, such as HTTPS, to encrypt data transmitted between the simulation application and external resources. This protects against data interception and tampering during transit.*
4. ***Error Handling****: Implementing robust error handling and validation mechanisms to prevent data breaches and ensure system stability. Proper error handling helps identify and respond to potential security vulnerabilities and anomalies effectively.*
5. ***Regular Updates****: Keeping software libraries, frameworks, and dependencies up-to-date to mitigate security vulnerabilities. Regular updates ensure that the system remains resilient against evolving security threats and maintains overall integrity.*

These security measures collectively enhance the resilience of the GPS Toll-based System simulation project, ensuring that sensitive data is protected, access is controlled, communications are secure, errors are managed effectively, and the system remains updated to address potential security risks. By adhering to these practices, the project aims to provide a secure and reliable environment for simulating toll-based transportation systems.

RESULTS AND ANALYSIS:



SIMULATION OUTPUT :

The simulation of the GPS Toll-based System generates comprehensive outputs that provide insights into various aspects of simulated toll-based transportation systems:

1. ***Vehicle Paths**: Visualizes the movement of vehicles along predefined routes using interactive maps. Each vehicle's path is traced from its initial location to its destination, illustrating simulated traffic patterns and flow dynamics.*
2. ***Toll Zones**: Defines virtual toll zones as polygonal areas on the map, based on geographical coordinates. These zones mark areas where toll charges are applicable, allowing for precise calculation of toll fees when vehicles enter these zones.*
3. ***Toll Charges**: Dynamically calculates toll charges based on the distance traveled by vehicles and the prevailing congestion levels. Different congestion levels (low, medium, high) influence the toll rates applied to vehicles passing through toll zones, reflecting real-world pricing strategies.*
4. ***Account Balances**: Tracks virtual user accounts associated with each vehicle throughout the simulation. It records initial account balances, deductions made for toll charges, and final balances after deducting these charges. This data provides insights into the financial impacts of toll charges on simulated users and allows for analysis of user behavior under different pricing scenarios.*

These outputs collectively enable stakeholders and operators to evaluate traffic management strategies, assess the financial implications of toll collection, and optimize system performance based on simulated data. By visualizing vehicle paths, defining toll zones, calculating dynamic toll charges, and monitoring account balances, the simulation provides a comprehensive overview of simulated toll-based transportation systems.

VISUALIZATION OF VEHICLE PATHS AND TOLL ZONES:

The visualization of vehicle paths and toll zones using Folium in the GPS Toll-based System simulation project enhances the understanding and analysis of simulated transportation dynamics:

1. ***Display Vehicle Paths**: Utilizes polylines to illustrate the routes taken by each vehicle from its starting point to its destination. Each vehicle's path is represented with distinct colors, enabling clear differentiation between routes and facilitating visual analysis of traffic flow.*
2. ***Highlight Toll Zones**: Represents virtual toll zones on the interactive map as filled polygons. These polygons visually delineate areas where toll charges are applicable, providing a spatial representation of toll boundaries. This visual clarity helps stakeholders identify and analyze the geographical impact of toll zones within the simulation environment.*
3. ***Visualize Crossings**: Marks intersections (crossings) between vehicle paths and toll zones using markers or circles on the map. These visual indicators highlight specific points where vehicles enter or exit defined toll zones, offering real-time feedback on traffic interactions with toll infrastructure.*

By integrating these visual elements into interactive maps generated by Folium, the simulation project enhances the visualization of vehicle movements, toll zone boundaries, and interactions between vehicles and toll infrastructure. This visual representation aids in evaluating traffic patterns, assessing toll collection efficiency, and optimizing transportation management strategies within the simulated environment.

TOLL CHARGES AND ACCOUNT BALANCES:

Toll charges and account balances play pivotal roles in assessing the financial dynamics and operational efficiency of the toll-based system simulation:

1. ***Toll Charges Calculation**: The simulation dynamically computes toll charges by considering two primary factors: distance traveled by vehicles within designated toll zones and the current congestion level. Congestion levels (low, medium, high) influence toll rates, impacting the financial burden on users. This dynamic calculation ensures that toll charges reflect real-time traffic conditions, optimizing revenue generation while managing traffic flow.*
2. ***Account Balances Management**: Throughout the simulation, virtual user accounts are meticulously tracked and updated. Initially, each account is endowed with a balance, which is subsequently adjusted based on deducted toll charges. This process mirrors real-world financial transactions, providing insights into the economic implications of toll fees on individual users. By monitoring account balances, the simulation facilitates analysis of user behavior and financial impacts under varying toll pricing strategies.*

These metrics—toll charges and account balances—serve as critical indicators for evaluating the financial sustainability, user acceptance, and operational effectiveness of the simulated toll-based system. They enable stakeholders to make informed decisions regarding

toll pricing policies, traffic management strategies, and system optimizations within the simulated environment.

ANALYSIS OF CONGESTION IMPACT ON TOLL CHARGES:

The analysis of congestion's impact on toll charges within the GPS Toll-based System simulation project focuses on several key aspects to optimize traffic management and revenue generation:

1. ***Congestion Levels**: Traffic congestion is categorized into predefined levels—low, medium, and high—based on simulated traffic density and flow patterns. These classifications help in identifying areas where congestion management measures are most needed.*
2. ***Toll Charge Variation**: Toll charges dynamically adjust in response to varying congestion levels. Higher congestion levels typically lead to increased toll rates within affected areas. This pricing strategy aims to regulate traffic flow, mitigate congestion, and optimize roadway usage.*
3. ***User Behavior and Responses**: The simulation analyzes how users respond to fluctuating toll charges influenced by congestion levels. Higher toll rates in congested areas may incentivize users to adjust their travel times, choose alternative routes, or use public transportation. Understanding these behavioral responses helps in predicting traffic distribution patterns and improving overall system efficiency.*

By examining these factors, the simulation project provides insights into the interplay between congestion, toll pricing strategies, and user behavior. This analysis supports informed decision-making for enhancing traffic management, optimizing revenue generation, and improving user experience within the simulated toll-based transportation system.

DISCUSSION

CHALLENGES FACED:

During the development and execution of the GPS Toll based System simulation project, several challenges emerged that required careful consideration and strategic solutions:

1. ***Geospatial Data Handling**: Managing and processing geospatial data, such as defining polygonal toll zones and calculating distances using GPS coordinates, presented challenges in algorithm implementation and data accuracy. Robust algorithms, including the Haversine formula for distance calculation, were essential to ensure precise simulation outcomes.*
2. ***Simulation Accuracy**: Ensuring the accuracy of vehicle movement simulations and toll charge calculations was crucial. Dynamic adjustments of toll rates based on simulated traffic conditions and congestion levels required sophisticated modeling and validation against real-world data to maintain fidelity in simulation results.*
3. ***Integration of UI Components**: Integrating diverse UI components—interactive maps (Folium), graphical charts (Matplotlib), and data tables (Tkinter ttk)—into a cohesive user interface posed challenges in terms of design consistency and functional integration. Aligning these components to provide seamless interaction and*

meaningful visualization of simulation outputs demanded meticulous design and testing.

4. ***Performance Optimization**: Optimizing the simulation's performance, particularly when scaling up the number of vehicles, routes, and toll zones, was critical. Balancing computational efficiency with simulation accuracy involved optimizing algorithms, data structures, and processing workflows to handle increasing complexities without compromising simulation fidelity.*

Addressing these challenges required a collaborative approach, leveraging expertise in geospatial data analysis, simulation modeling, UI/UX design, and performance optimization. By overcoming these obstacles, the GPS Toll based System simulation project aims to provide a robust platform for studying and optimizing toll-based transportation systems effectively.

LIMITATIONS OF THE SYSTEM:

Despite its capabilities, the GPS Toll based System simulation project acknowledges several limitations that impact its representation of real-world scenarios:

1. ***Simplified Traffic Dynamics**: The simulation simplifies traffic behavior and congestion patterns, which may not fully capture the complexities of real-world traffic dynamics, including driver behaviors, unpredictable events (like accidents), and dynamic traffic flow changes.*
2. ***Static Toll Zone Definition**: Toll zones are defined as static polygonal shapes based on initial configurations. This static approach does not account for real-time adjustments in toll boundaries or dynamic changes in traffic conditions that could affect optimal toll zone placements.*
3. ***Single User Simulation**: Each vehicle operates independently with predefined routes and behaviors, lacking interaction or coordination among vehicles. This limitation excludes the influence of vehicle-to-vehicle interactions, cooperative driving behaviors, or coordinated traffic management strategies.*
4. ***Scope of Congestion Modeling**: Congestion levels are categorized into predefined levels (low, medium, high), without mechanisms for dynamic adaptation based on real-time simulation outcomes. This restricts the simulation's ability to learn from traffic patterns and adjust congestion levels dynamically.*

These limitations highlight areas for potential enhancements and future developments in the GPS Toll based System simulation project. Addressing these challenges could involve incorporating more advanced traffic simulation models, integrating real-time data feeds for dynamic toll adjustments, and enhancing vehicle interaction and coordination within the simulation environment. By expanding its scope and refining its methodologies, the project aims to better simulate and optimize toll-based transportation systems with increased accuracy and realism.

FUTURE ENHANCEMENTS:

To enhance the GPS Toll based System simulation and address its challenges, several advanced features and enhancements can be implemented:

1. ***Real-time Traffic Simulation**:*
    - *Implement advanced algorithms to simulate real-time traffic dynamics. This includes adaptive traffic flow models that adjust based on current conditions such as traffic density, speed limits, and road capacities.*
    - *Incorporate vehicle interactions and cooperative behaviors to simulate realistic traffic scenarios where vehicles respond dynamically to each other's movements, lane changes, and traffic signals.*
2. ***Dynamic Toll Zone Management**:*
    - *Introduce dynamic toll zone definitions that adapt in real-time based on traffic conditions and congestion levels. Utilize real-time data feeds from sensors, GPS tracking, and predictive analytics to optimize toll collection points and tariff adjustments.*
    - *Implement algorithms to dynamically adjust toll rates within zones based on current traffic flow, encouraging traffic redistribution and congestion alleviation strategies.*
3. ***Multi-agent Simulation**:*
    - *Enhance the simulation framework to support multi-agent interactions among vehicles. Vehicles should dynamically adjust routes based on real-time traffic information, congestion levels, and toll charges.*
    - *Enable vehicles to communicate and cooperate within the simulation, allowing for coordinated lane changes, merging maneuvers, and route optimizations based on shared traffic information.*
4. ***User Behavior Modeling**:*
    - *Develop behavioral models to simulate diverse user responses to toll charges, traffic conditions, and alternative route options.*
    - *Model user preferences, such as willingness to pay higher tolls for faster routes or preferences for less congested paths, to simulate realistic decision-making processes among drivers.*
5. ***Integration of External Data Sources**:*
    - *Integrate real-time data sources such as traffic APIs, weather updates, and road condition reports to enhance simulation accuracy.*
    - *Utilize external data to simulate the impact of weather conditions, accidents, road closures, and special events on traffic patterns and toll collection operations.*

By incorporating these enhancements, the GPS Toll based System simulation can evolve into a more robust and realistic tool for evaluating toll-based transportation systems. These features not only improve accuracy and realism but also enable the simulation to provide actionable insights for optimizing traffic management, enhancing user experience, and maximizing revenue generation in toll operations.

SUMMARY OF ACHIEVEMENTS:

The GPS Toll based System simulation project has achieved several significant milestones, showcasing its capabilities in simulating and analyzing toll-based transportation systems:

1. ***Accurate Simulation**:*
    - *Successfully simulated vehicle movements along predefined routes using GPS coordinates. The project accurately calculates distances traveled and dynamically computes toll charges within defined toll zones based on simulated traffic conditions.*
2. ***Visualization Capabilities**:*
    - *Developed interactive maps using Folium and graphical representations with Matplotlib to visualize key aspects such as vehicle paths, toll zones, and toll charges. These visualizations provide intuitive insights into traffic patterns, congestion hotspots, and toll impacts across simulated routes.*
3. ***User Interface Integration**:*
    - *Integrated a user-friendly interface using Tkinter, facilitating interactive display of simulation outputs. Users can easily visualize and analyze vehicle movements, toll charges, account balances, and congestion analysis through intuitive graphical representations and data tables.*
4. ***Comprehensive Analysis**:*
    - *Conducted thorough analysis of simulation results to evaluate the impact of congestion on toll charges and user behavior responses. The project provides insights into traffic management strategies by analyzing system performance metrics and user interactions with toll pricing dynamics.*

These achievements highlight the project's ability to simulate realistic scenarios, visualize complex data effectively, and support informed decision-making in optimizing toll-based transportation systems. The integration of advanced simulation techniques, visualization tools, and user-friendly interfaces contributes to its success in addressing challenges and enhancing the understanding of traffic dynamics and toll operations.

PROJECT SIGNIFICANCE:

The GPS Toll based System simulation project plays a crucial role in advancing urban transportation and toll collection systems in several key areas:

1. ***Traffic Management Optimization**:*
    - *The project facilitates the optimization of traffic flow and congestion management strategies. By dynamically adjusting toll pricing based on real-time simulation data, it helps in reducing congestion, improving traffic efficiency, and enhancing overall mobility within urban areas.*
2. ***Toll Collection Efficiency**:*
    - *It enhances the efficiency of toll collection systems by accurately calculating toll charges and managing user accounts in simulated environments. This capability supports effective revenue generation, ensuring sustainable funding for maintaining transportation infrastructure and services.*

3. ***Policy Evaluation***:
    - *The simulation project provides valuable insights for policymakers and urban planners. It aids in evaluating and refining toll policies, assessing their socio-economic impacts, and optimizing investments in transportation infrastructure. This data-driven approach helps in making informed decisions to support urban development and mobility planning.*
4. ***Educational and Research Use***:
    - *As an educational tool, the project offers a platform for studying geographical algorithms, traffic simulations, and toll collection mechanisms. It serves academia and researchers by providing a simulated environment to explore urban mobility challenges, test hypotheses, and develop innovative solutions for future transportation systems.*

Overall, the GPS Toll based System simulation project contributes significantly to advancing urban transportation management, improving toll collection efficiency, informing policy decisions, and fostering research and education in the field of urban mobility and infrastructure planning.

CONCLUSION:

The GPS Toll based System simulation project represents a significant leap forward in simulating and analyzing toll-based transportation systems. Leveraging Python programming and GIS techniques alongside interactive visualization tools like Folium and Matplotlib, the project has achieved several milestones:

1. ***Accurate Simulation and Calculation***:
    - *Successfully modeled vehicle movements along predefined routes using GPS coordinates, ensuring precise calculation of toll charges based on distance traveled and congestion levels. This accuracy is crucial for evaluating toll policies and optimizing traffic management strategies.*
2. ***User-Friendly Interface***:
    - *Developed a user-friendly interface using Tkinter to visualize simulation outputs effectively. This interface enhances usability, allowing stakeholders to interpret traffic dynamics, toll charges, and congestion impacts intuitively.*
3. ***Dynamic Toll Zone Management***:
    - *Integrated dynamic toll zone definitions that adjust in real-time based on traffic conditions. This capability provides insights into optimizing traffic flow and managing congestion effectively, contributing to enhanced toll collection efficiency.*
4. ***Practical Applications and Significance***:
    - *Beyond technical implementation, the project supports practical applications in traffic management, policy evaluation, and urban planning. It serves as a critical tool for policymakers and researchers to analyze toll policies, assess socio-economic impacts, and inform decisions related to urban mobility solutions.*

Looking ahead, the project holds potential for further advancements:

- ***Real-Time Data Integration**: Integrating real-time data sources such as traffic APIs and weather data to enhance simulation accuracy and responsiveness to external factors.*
- ***Multi-Agent Simulations**: Enhancing simulations to support interactions among vehicles, allowing for dynamic route adjustments based on real-time traffic information.*
- ***Advanced Analytics**: Incorporating advanced analytics for predictive traffic management, enabling proactive measures to mitigate congestion and optimize traffic flow.*

In conclusion, the GPS Toll based System simulation project exemplifies innovation in traffic engineering, promising improvements in transportation infrastructure efficiency, user experience, and sustainable urban development. It sets a precedent for future advancements in toll-based transportation systems worldwide, driven by technology-driven solutions and data-driven insights.

REFERENCES:

The GPS Toll based System simulation project has drawn upon various sources and documentation to inform its development and implementation. Key references include:

1. Python Documentation - Official Python programming language documentation. Available at: https://docs.python.org

2. Tkinter Documentation - Official documentation for Tkinter GUI toolkit. Available at: https://docs.python.org/3/library/tk.html

3. Matplotlib Documentation - Official documentation for Matplotlib plotting library. Available at: https://matplotlib.org/stable/contents.html

4. Folium Documentation - Official documentation for Folium, a Python library for creating interactive maps. Available at: https://pythonvisualization.github.io/folium/ Untitled 24

5. Geographic Information System (GIS) Resources - Resources and guides on handling geographic data and calculations, including Haversine formula. Various online sources and GIS documentation.

6. Traffic Simulation and Congestion Modeling - Academic papers, articles, and simulation methodologies related to traffic flow modeling, congestion impact analysis, and toll collection systems.

7. Urban Mobility and Transportation Studies - Research papers, reports, and case studies on urban transportation infrastructure, toll policies, and traffic management strategies.

These references have been instrumental in shaping the methodology, implementation, and analysis of the GPS Toll based System simulation project, providing foundational knowledge and technical guidance throughout its development.

APPENDICES:

Detailed Code Listing

1. ***Class Definitions (Point, Polygon):***

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __repr__(self):
        return f"Point({self.x}, {self.y})"

class Polygon:
    def __init__(self, vertices):
        self.vertices = vertices

    def contains(self, point):
        # Insert contains method code here
        pass
```

2. ***Functions(Haversine,move_vehicle,check_toll_zone_crossings, calculate_dynamic_toll):***

```
def haversine(coord1, coord2):
    # Insert Haversine function code here
    pass

def move_vehicle(vehicle, vehicle_paths, steps=10):
    # Insert move_vehicle function code here
    pass

def check_toll_zone_crossings(vehicle_path, toll_zones):
    # Insert check_toll_zone_crossings function code here
    pass

def calculate_dynamic_toll(congestion_level, dista nce_travelled):
    # Insert calculate_dynamic_toll function code here
    pass
```

GLOSSARY OF TERMS :

- ***GPS:*** *Global Positioning System, a satellite-based navigation system.*
- ***GIS:*** *Geographic Information System, a system for capturing, storing, analyzing, and managing geographical data.*
- ***Toll Zone:*** *A defined area where toll charges are applicable for vehicles passing through. Congestion: Traffic congestion refers to the condition on road networks that occurs when traffic demand exceeds the available capacity.*
- ***Haversine Formula:*** *A formula used to calculate the shortest distance between two points on the surface of a sphere, given their latitudes and longitudes.*
- ***Simulation:*** *The imitation of a real-world process or system over time.*