INTEL UNNATI INDUSTRIAL TRAINING PROGRAM 2024

PROJECT REPORT
ON

GPS TOLL BASED SYSTEM SIMULATION IN PYTHON

MEMBERS

NAME	ROLL No.
S. Ayush Prusty	2229148
Priyanshu Ranjan	2229139
Piyush Verma	2229136
Ritoban Ghosh	2229145

INSTITUTE:- Kalinga Institute of Industrial Technology

INTRODUCTION

The number of vehicles on the road are constantly on the rise. This in turn will lead to a greater number of vehicles on the road. It would be quite a challenge to monitor the movement of these vehicles on the road. In today's fast-paced world, efficient transportation systems are crucial for economic growth and societal well-being. One critical aspect of transportation infrastructure is toll collection. Whether it's a highway, bridge, or tunnel, tolls play a significant role in funding and maintaining these essential routes.

The purpose of our project is to design and implement an advanced toll collection system that enhances efficiency, accuracy, and user experience..The project aims at addressing such problems which the government or any other organization might encounter. This project contains a simulation which has been designed using Python. The simulation will help to calculate the charges which have been incurred by a user while commuting on a toll road. The toll is charged on the basis of the actual distance which has been travelled on the tolled road. The toll would be deducted from the user's account. By automating payments and minimizing wait times, we empower commuters to focus on their journey rather than the toll booth ahead.

Beyond the obvious revenue generation, tolls serve as bridges—both literal and metaphorical. They bridge the gap between infrastructure costs and maintenance, ensuring that our roads, bridges, and tunnels remain safe and functional. By leveraging technology, we aim to streamline the toll payment process.

OBJECTIVES

1) To leverage python in order to create an easy to use toll collection system.
2) To simplify the toll payment system.
3) To make the system more user friendly.

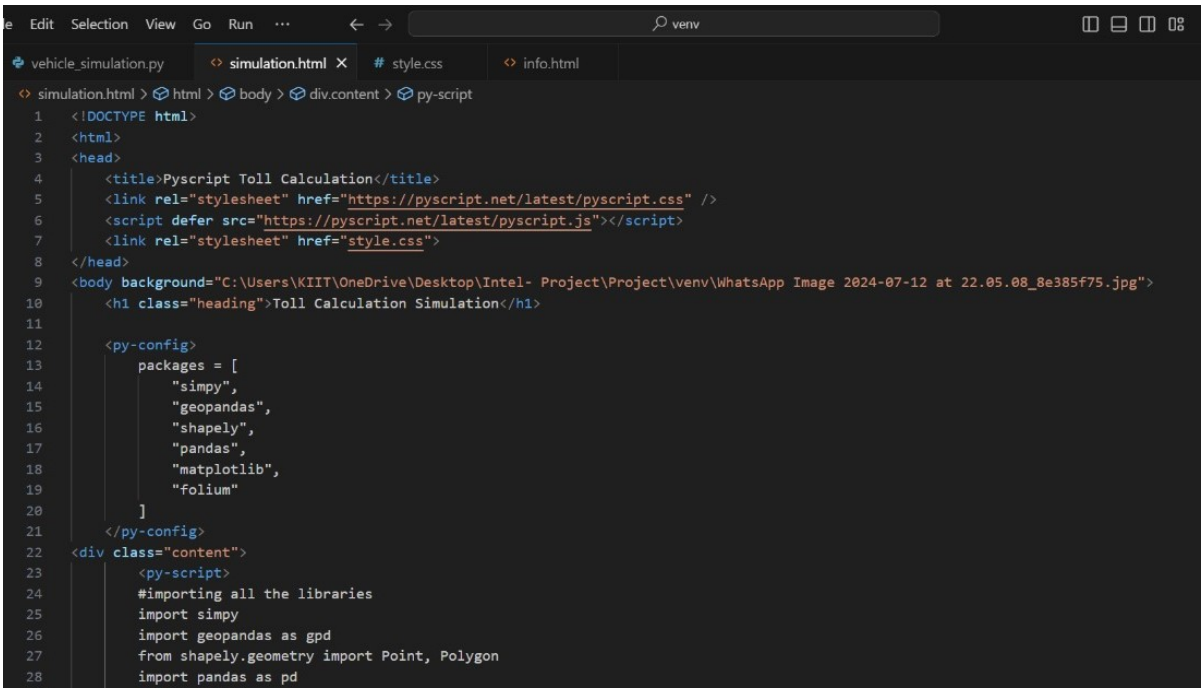
RESOURCES USED

- **WEB PAGE-**
  1. HTML for web page designing
  2. CSS for web page styling

- **PYTHON-**
  1. Simpy to stimulate vehicle movement
  2. Pandas for handling data related to vehicles.
  3. Shapely for defining the toll zones.
  4. Haversine for distance calculation between points on the map.
  5. Matplotlib and Folium for visualization of the simulation.

METHODOLOGY

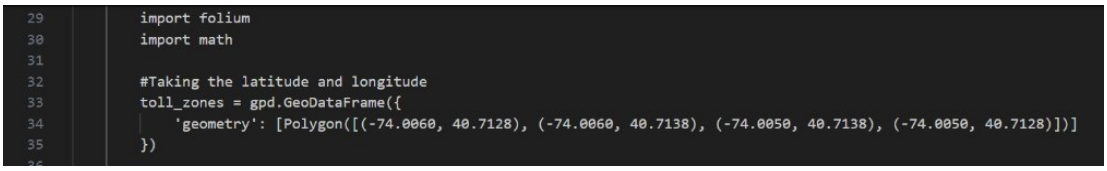
1. Determining all the use cases.
2. The resources required for the simulation.



```
<!DOCTYPE html>
<html>
<head>
    <title>Pyscript Toll Calculation</title>
    <link rel="stylesheet" href="https://pyscript.net/latest/pyscript.css" />
    <script defer src="https://pyscript.net/latest/pyscript.js"></script>
    <link rel="stylesheet" href="style.css">
</head>
<body background="C:\Users\KIIT\OneDrive\Desktop\Intel- Project\Project\venv\WhatsApp Image 2024-07-12 at 22.05.08_8e385f75.jpg">
    <h1 class="heading">Toll Calculation Simulation</h1>

    <py-config>
        packages = [
            "simpy",
            "geopandas",
            "shapely",
            "pandas",
            "matplotlib",
            "folium"
        ]
    </py-config>
<div class="content">
        <py-script>
        #importing all the libraries
        import simpy
        import geopandas as gpd
        from shapely.geometry import Point, Polygon
        import pandas as pd
```

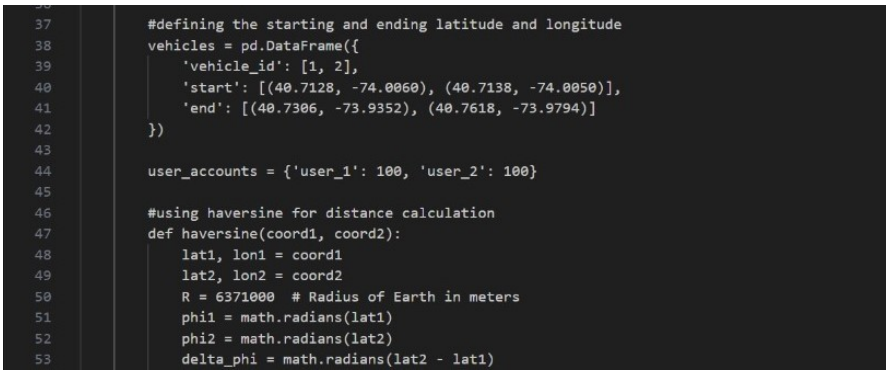
3. Firstly the toll road was determined where the user would be charged.



```
        import folium
        import math

        #Taking the latitude and longitude
        toll_zones = gpd.GeoDataFrame({
            'geometry': [Polygon([(-74.0060, 40.7128), (-74.0060, 40.7138), (-74.0050, 40.7138), (-74.0050, 40.7128)])]
        })
```

4. Then the vehicles are assigned id's which would make it easier to know about their whereabouts.
5. Determining the start and end positions of the the given vehicle.



```
        #defining the starting and ending latitude and longitude
        vehicles = pd.DataFrame({
            'vehicle_id': [1, 2],
            'start': [(40.7128, -74.0060), (40.7138, -74.0050)],
            'end': [(40.7306, -73.9352), (40.7618, -73.9794)]
        })

        user_accounts = {'user_1': 100, 'user_2': 100}

        #using haversine for distance calculation
        def haversine(coord1, coord2):
            lat1, lon1 = coord1
            lat2, lon2 = coord2
            R = 6371000  # Radius of Earth in meters
            phi1 = math.radians(lat1)
            phi2 = math.radians(lat2)
            delta_phi = math.radians(lat2 - lat1)
```

6. Determine whether the vehicle crosses the toll zone or not.
7. Update the position of the vehicle.
8. Set the toll rate which would be levied.

```
        delta_phi = math.radians(lat2 - lat1)
        delta_lambda = math.radians(lon2 - lon1)
        a = math.sin(delta_phi / 2) ** 2 + math.cos(phi1) * math.cos(phi2) * math.sin(delta_lambda / 2) ** 2
        c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
        return R * c

    #final total toll calculation
    def calculate_toll(distance):
        rate_per_km = 10.0  # Example rate
        return distance / 1000 * rate_per_km

    #toll amount calculation
    def deduct_toll(user, amount):
        user_accounts[user] -= amount

    #checkin toll intersection
    def check_toll_crossing(vehicle_position):
        point = Point(vehicle_position[::-1])
        for _, toll_zone in toll_zones.iterrows():
            if toll_zone['geometry'].contains(point):
                return True
        return False
```

9. Calculate the toll which is to be charged on the basis of the distance travelled.
10. Cut the required amount from the user accounts.
11. Update the bank balance of the users.

```
        vehicle_movements = []

        def vehicle(env, vehicle_id, user, start, end, speed):
            position = start
            distance = haversine(start, end)
            travel_time = distance / speed
            toll_distance = 0

            for _ in range(int(travel_time)):
                yield env.timeout(1)
                position = (position[0] + 0.0001, position[1] + 0.0001)
                if check_toll_crossing(position):
                    toll_distance += haversine(start, position)
                    start = position

            toll_charge = calculate_toll(toll_distance)
            deduct_toll(user, toll_charge)
            vehicle_movements.append({
                'vehicle_id': vehicle_id,
                'user': user,
                'toll_distance': toll_distance,
                'toll_charge': toll_charge
            })
            print(f'Updated {user} balance : Rs{user_accounts[user]:.2f}')
```

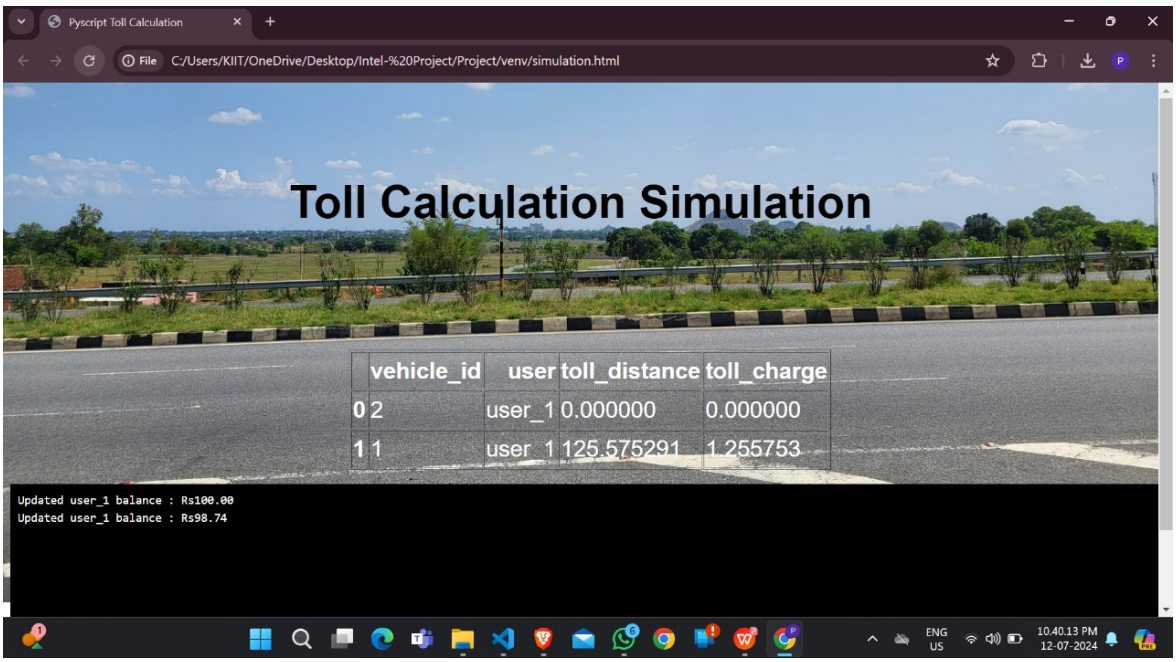
```
        def run_simulation():
            env = simpy.Environment()
            for index, row in vehicles.iterrows():
                env.process(vehicle(env, row['vehicle_id'], 'user_1', row['start'], row['end'], speed=10))
            env.run()

            report = pd.DataFrame(vehicle_movements)
            display(report)

            map = folium.Map(location=[40.7128, -74.0060], zoom_start=12)
            for _, toll_zone in toll_zones.iterrows():
                folium.GeoJson(toll_zone['geometry']).add_to(map)
            for _, row in vehicles.iterrows():
                folium.Marker(location=row['start'], popup='Start').add_to(map)
                folium.Marker(location=row['end'], popup='End').add_to(map)
            map.save('map.html')

        run_simulation()
    </py-script>
    </p>
</div>
</body>
</html>
```

12. Provide all the relevant information to the user through a web page.



SCOPE OF IMPROVEMENT

1. Different toll rates can be defined for different kinds of vehicles (heavy and light vehicles).
2. The payment system can be made more secure.
3. Dynamic pricing(Congestion based,Time slot based) can be incorporated.
4. Vehicle movement reports can be generated.

CONCLUSION

In conclusion, our project addresses the growing challenge of monitoring and managing the increasing number of vehicles on the road. By designing and implementing an advanced toll collection system, we aim to enhance efficiency, accuracy, and user experience in toll payment processes. The Python-based simulation effectively calculates toll charges based on the actual distance traveled, ensuring fair and precise billing. Automating payments and reducing wait times not only improve commuter convenience but also contribute to smoother traffic flow.

Moreover, the project underscores the critical role of tolls in funding and maintaining transportation infrastructure. By bridging the gap between infrastructure costs and maintenance, our system ensures that roads, bridges, and tunnels remain safe and functional. Leveraging technology to streamline toll payments, we pave the way for a more efficient and user-friendly transportation system, ultimately supporting economic growth and societal well-being.