Name: Bavadharani B

REG no:113323106009

DEPT:ECE

Nm Id:aut113323ecb04

Phase 3: Implementation of Project

Title: Traffic Flow Optimization

Objective

To implement a smart traffic flow optimization system using AI and IoT technologies. This phase focuses on real-time traffic analysis, AI-based signal control, basic IoT sensor integration, and data privacy protocols.

1. AI Model Development

Overview

The primary feature of the AI-Powered Healthcare Assistant is its ability to assess user symptoms and provide health-related recommendations. In Phase 3, the AI model will be trained and implemented to recognize basic health issues.

Implementation

1.Computer Vision Model: AI is trained to process traffic camera feeds to detect vehicle density and flow rates.

2. Data Source: Historical traffic data and simulated input are used to train the model for accurate predictions.

Outcome

By the end of this phase, the AI model will adjust signal durations in real time based on traffic volume at intersections.

---

2. Smart Signal Control Interface

Overview:
Traffic signals will be managed through an intelligent interface that interacts with the AI model.

Implementation:

- **Signal Automation:** The interface receives input from the AI to adjust green/red light durations.

- **User Dashboard:** A control panel allows traffic officers to override AI decisions when needed.

Outcome:
Functional integration between AI and traffic lights ensures smoother traffic flow and reduced congestion.

3. IoT Sensor Integration (Optional)

Overview:
IoT sensors will collect real-time data from roads and vehicles to supplement the AI's decision-making.

Implementation:

- **Sensors:** Pressure sensors, RFID, and vehicle counters deployed at intersections.
- **APIs:** Integration with GPS and vehicle telemetry systems for more precise inputs.

Outcome:
The framework for smart sensor input is established, enhancing the AI’s adaptability to real-world conditions.

4. Data Security Implementation

Overview:
Traffic and vehicle data collected through sensors and cameras must be handled securely.

Implementation:

- **Encryption:** All traffic data transmissions are encrypted.
- **Storage:** Data is stored in a secure, access-controlled environment to protect user identity and traffic history.

Outcome:
Robust protection measures ensure compliance with smart city data regulations.

5. Testing and Feedback Collection

Overview:
Initial field testing is conducted to evaluate signal efficiency and traffic throughput.

Implementation:

- **Pilot Testing:** Selected intersections are monitored to analyze AI decision outcomes.
- **Feedback Loop:** Traffic police and commuters provide input on effectiveness and response time.

Outcome:
Test results guide further optimization and scaling in the next development phase.

Challenges and Solutions

1. **AI Accuracy**
    - *Challenge:* Inconsistent camera feed or sensor data.
    - *Solution:* Use redundancy and machine learning feedback loops for continuous improvement.
2. **User Experience**
    - *Challenge:* Manual overrides might conflict with AI decisions.
    - *Solution:* Develop clear alert protocols and override conditions.
3. **Sensor Availability**
    - *Challenge:* Limited infrastructure in some zones.
    - *Solution:* Simulate inputs or prioritize high-traffic zones for initial deployment.

Outcomes of Phase 3

1. Real-time AI Traffic Management
2. Dynamic Signal Control System
3. Optional IoT Integration Framework
4. Secure Data Handling
5. Initial Deployment and Performance Evaluation

Next Steps for Phase 4

1. Expand to additional intersections and roads.
2. Integrate with public transport and emergency vehicle tracking.
3. Enhance the AI model using continuous real-time data.

SCREENSHOTS OF CODE

main.py

Share Run

```
import pandas as pd

# Sample data: traffic inflow (vehicles per minute) at a 4-way intersection
data = {
    'Direction': ['North', 'South', 'East', 'West'],
    'Vehicles_Per_Minute': [20, 25, 15, 30],
}

df = pd.DataFrame(data)

# Total vehicles per minute
total_vehicles = df['Vehicles_Per_Minute'].sum()

# Calculate green light time based on vehicle flow ratio
df['Green_Light_Seconds'] = (df['Vehicles_Per_Minute'] / total_vehicles) * 120  # 120s total cycle time

# Optimize: sort directions by highest traffic
df_sorted = df.sort_values(by='Vehicles_Per_Minute', ascending=False)

print("Traffic Flow Optimization:")
print(df_sorted.to_string(index=False))
```

Output

```
Traffic Flow Optimization:
Direction  Vehicles_Per_Minute  Green_Light_Seconds
     West                   30            40.000000
    South                   25            33.333333
    North                   20            26.666667
     East                   15            20.000000

=== Code Execution Successful ===
```