Public Tansport optimization

Developing an IOT based python script for public Tansport optimization

Hardware setup:

Choose appropriate sensors (e.g., RFID, GPS, cameras, or occupancy sensors) for passenger monitoring.

Set up the sensors in public transport vehicles to collect data on passenger boarding, alighting, and occupancy.

Data Collection:

Write code to collect data from the sensors in real-time.

This data might include information about passenger count, their entry and exit points, and vehicle location.

Data Processing:

Process the collected sensor data to extract relevant information.

Filter and clean the data to remove noise and errors.

Passenger Monitoring:

Use the processed data to monitor passenger movements and occupancy levels in real-time.

Optimization Algorithms:

Implement optimization algorithms to improve the efficiency of public transport services.

Consider factors like minimizing wait times, optimizing routes, and maximizing vehicle occupancy.

User Interface:

Create a user interface for both transportation operators and passengers.

Passengers can check real-time information, while operators can make data-driven decisions.

Machine Learning and Predictive Analytics:

Impement machine learning models for demand prediction and route optimization.

Use historical data to make predictions about future passenger demand.

Alerts and Notifications:

Implement alerts and notifications for passengers, such as estimated arrival times and disruptions.

Data Storage and Analysis:

Store historical data in a database for further analysis.

Use tools like pandas and NumPy for in-depth data analysis.

Reporting and Visualization:

Create dashboards or reports to visualize key performance metrics and insights.

Testing and Validation:

Rigorously test the system under different scenarios to ensure its reliability and accuracy.

Deployment:

Deploy the system on public transport vehicles and in transportation control centers.

Security and privacy:

Implement security measures to protect passenger data and ensure privacy compliance.

Maintenance and Updates:

Regularly maintain and update both the hardware and software components of the system.

Python script:

```
class PublicTransport:
    def __init__(self, max_capacity):
        self.max_capacity = max_capacity
        self.passenger_count = 0


    def board_passenger(self, count=1):
        if self.passenger_count + count <= self.max_capacity:
            self.passenger_count += count
            print(f"Boarded {count} passenger(s). Total passengers: {self.passenger_count}")
        else:
            print("The vehicle is already at maximum capacity. Cannot board more passengers.")

    def alight_passenger(self, count=1):
        if self.passenger_count - count >= 0:
            self.passenger_count -= count
            print(f"Alighted {count} passenger(s). Total passengers: {self.passenger_count}")
        else:
            print("There are no passengers to alight from the vehicle.")

if __name__ == "__main__":
    public_transport = PublicTransport(max_capacity=60)

    while True:
        print("\n1. Board passengers")
        print("2. Alight passengers")
        print("3. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            count = int(input("Enter the number of passengers to board: "))
            public_transport.board_passenger(count)
```

```
elif choice == "2":
    count = int(input("Enter the number of passengers to alight: "))
    public_transport.alight_passenger(count)
elif choice == "3":
    print("Exiting the program.")
    break
else:
    print("Invalid choice. Please select a valid option.")
```