Public Transport Optimization

Submitted By:

Sriram V
(au80421205070)

Introduction:

Public transport optimization in the context of the Internet of Things (IoT) refers to the use of IoT technologies and data analytics to improve the efficiency, safety, and overall quality of public transportation systems.

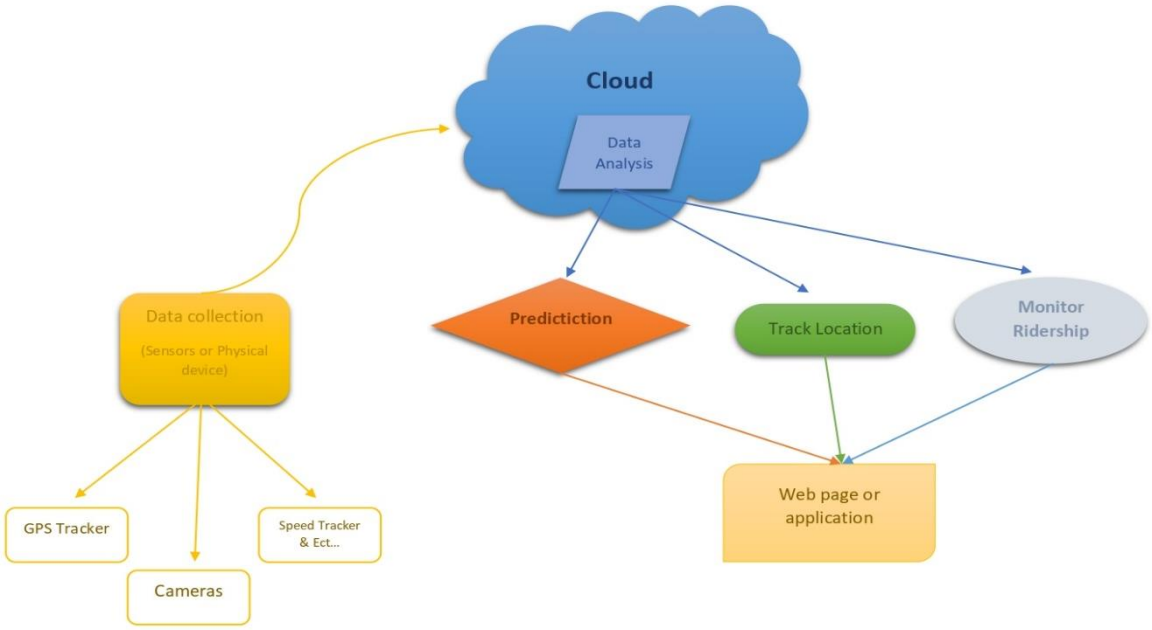
Project Objective:

Public transport optimization in the context of the Internet of Things (IoT) refers to the use of IoT technologies and data analytics to improve the efficiency, arrival prediction, and overall quality of public transportation.

Project Definition:

The project involves integrating IoT sensors into public transportation vehicles to monitor ridership, track locations, and predict arrival times. The goal is to provide real-time transit information to the public through a public platform, enhancing the efficiency and quality of public transportation services. This project includes defining objectives, designing the IoT sensor system, developing the real-time transit information platform, and integrating them using IoT technology and Python.

Design for Problem:



Project Steps:

1. IOT Devices and Data Collection:

IoT devices play a pivotal role in enhancing public transport optimization through efficient data collection.

Sensors for Deployment:

To achieve effective public transport optimization, several types of sensors can be deployed in vehicles and at transport infrastructure locations. Here are the key sensors and their applications.

❖ GPS Sensor:

Application: Used for real-time vehicle tracking and route optimization.

Benefits: Provides accurate location data, helping to monitor vehicle movement, calculate ETA, and optimize routes based on traffic conditions.

❖ Passenger Counting Sensors(camera):

Application: Used to monitor passenger loads on vehicles.

Benefits: Allows for optimization of vehicle capacity, leading to better resource allocation and service planning.

❖ Temperature and Humidity Sensor:

Application: Monitoring and maintaining comfortable climate conditions inside vehicles.

Benefits: Ensures passenger comfort and safety by regulating heating, ventilation, and air conditioning systems.

❖ Proximity Sensors (Ultrasonic or Infrared):

Application: Detecting the proximity of vehicles to obstacles, objects, or pedestrians.

Benefits: Enhances safety by providing alerts to drivers and helping avoid collisions.

❖ Camera:

Application: Surveillance and monitoring of passengers, driver behaviour, and security.

Benefits: Improves safety and security by recording video footage for analysis and incident resolution.

Raspberry Pi

The Raspberry Pi is a low cost. Credit-card sized computer that plugs into a computer monitor or TV and uses a standard keyboard and mouse.it is a capable little device that enables people of all ages to explore computing and to learn how to program in languages like scratch and python. Python is an easy to start with high-level language, and it is an integral part of the Raspberry Pi's operating system.

Here we used to collect real time data from IOT Devices like (GPS Sensor, Humidity Sensor, Camera, Ultrasonic Sensor) and transmit to cloud.

2. Data Storage:

Collect the real time information using IoT devices with sensors including GPS, passenger counts, temperature, proximity, and camera footage and transmitting to a Thing Speak cloud for data analyzing.

Python Script Development:

Create Python scripts to analyze and process sensor data, implementing algorithms for optimization, real-time passenger information, and more.

Install Required Libraries collecting data from devices:

You will need to install Python libraries to interact with the sensors and transmit data to the cloud. For example

- For GPS: serial
- For Temperature and Humidity: Adafruit_DHT
- For Passenger Count Sensor: RPi.GPIO as GPIO

Python Program:

```
import time

import requests

import Adafruit_DHT
```

```
import RPi.GPIO as GPIO
import serial
THING_SPEAK_API_KEY = 'YOUR_API_KEY'
THING_SPEAK_URL = f'https://api.thingspeak.com/update?api_key={THING_SPEAK_API_KEY}'
SENSOR_PIN = 17
passenger_count = 0
GPIO.setmode(GPIO.BCM)
GPIO.setup(SENSOR_PIN, GPIO.IN)
DHT_SENSOR = Adafruit_DHT.DHT22
DHT_PIN = 4  # GPIO pin where the DHT22 sensor is connected
SERIAL_PORT = '/dev/ttyS0'  # Raspberry Pi's serial port
ser = serial.Serial(SERIAL_PORT, baudrate=9600, timeout=1)
while True:
    if GPIO.input(SENSOR_PIN):
        passenger_count += 1
    humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
    gps_data = ser.readline().decode('utf-8')
    data = {
        'field1': temperature,
        'field2': humidity,
        'field3': passenger_count,
        'field4': gps_data,
    }

    try:
        response = requests.get(THING_SPEAK_URL, params=data)
        if response.status_code == 200:
            print("Data sent to ThingSpeak successfully.")
        else:
            print(f"Failed to send data to ThingSpeak. Status code: {response.status_code}")
```

```
except Exception as e:
    print(f"Error sending data to ThingSpeak: {str(e)}")

time.sleep(15)
```

3. Data Processing and Analysis:

Data processing and analysis in the context of public transport optimization is a critical component of improving transit systems. The collected data from IoT devices, such as GPS trackers and sensors on public transport vehicles, are processed to extract valuable insights. This process involves real-time monitoring of vehicle locations, passenger loads, and traffic conditions. Use the Analysis algorithm to get correct results.

Advanced data analytics and machine learning algorithms are applied to the collected data to gain insights and make informed decisions. This analysis can help transportation authorities and service providers optimize various aspects of the public transport system, such as scheduling, routing, and maintenance.

Prediction:

Use the Linear Regression algorithm to improve the arrival time based on historical data and traffic conditions.

Linear Regression:

Linear regression algorithm to improve arrival time prediction for public transport optimization involves several steps. Linear regression is a simple and widely used method for predicting a continuous target variable (in this case, arrival time) based on one or more input features (historical data and traffic conditions).

Use this equation to predict:

$$y = b0 + b1*x1 + b2*x2 + ... + bn*xn$$

Example algorithm to perform linear regression:

```
import numpy as np
class LinearRegression:
    def __init__(self, learning_rate=0.01, num_iterations=1000):
        self.learning_rate = learning_rate
        self.num_iterations = num_iterations
        self.weights = None
        self.bias = None
    def fit(self, X, y):
        num_samples, num_features = X.shape
        self.weights = np.zeros(num_features)
        self.bias = 0
        for _ in range(self.num_iterations):
            linear_model = np.dot(X, self.weights) + self.bias
            dw = (1/num_samples) * np.dot(X.T, (linear_model - y))
            db = (1/num_samples) * np.sum(linear_model - y)
            self.weights -= self.learning_rate * dw
            self.bias -= self.learning_rate * db
    def predict(self, X):
        linear_model = np.dot(X, self.weights) + self.bias
        return linear_model
if __name__ == "__main__":
X = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
y = np.array([4, 8, 12])
model = LinearRegression(learning_rate=0.01, num_iterations=1000)
model.fit(X, y)
predictions = model.predict(X)
print("Predicted values:", predictions)
```

4. Web Development:

Web development can be a powerful tool in the realm of public transport optimization, as it provides a user-friendly interface for both transit authorities and commuters. Through well-designed websites and mobile apps, passengers can access real-time information on bus or train schedules, route changes, and delays, making their daily commute more predictable and efficient.

Front-end:

HTML, CSS, and **JavaScript** will be used for the front-end development to create the user interface.

1. HTML/CSS: Create the structure and style of the platform, including maps, tables, charts, and real-time updates.

Code:

index.html

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>Public Transport Dashboard</title>
   <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
   <div class="header">
      <h1>Public Transport Dashboard</h1>
   </div>
   <div class="container">
      <div class="map">
         <div id="map"></div>
      </div>
      <div class="data">
         <div class="data-item">
            <h2>Real-time Location</h2>
            <p id="location">Loading...</p>
         </div>
         <div class="data-item">
            <h2>Ridership</h2>
            <p id="ridership">Loading...</p>
         </div>
         <div class="data-item">
            <h2>Arrival Time</h2>
            <p id="arrival-time">Loading...</p>
         </div>
      </div>
   </div>
   <script src="script.js"></script>
</body>
```

```
</html>
```

styles.css:

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

.header {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 15px;
}
.container {
    display: flex;
    margin: 20px;
}

.map {
    flex: 2;
    height: 400px;
    border: 1px solid #ccc;
}

.data {
    flex: 1;
    padding: 20px;
}

.data-item {
    margin-bottom: 20px;
}

h2 {
    margin: 0;
}

#map {
    width: 100%;
    height: 100%;
}
```

2. JavaScript: Use JavaScript to make the platform interactive and to handle real-time updates. You'll need to use libraries or frameworks like Leaflet for mapping and Chart.js for data visualization.

Code:
script.js

```
const staticData = {
   location: "City Center",
   ridership: 120,
   arrivalTime: "10 minutes",
};
document.getElementById("location").textContent = staticData.location;
document.getElementById("ridership").textContent = staticData.ridership;
document.getElementById("arrival-time").textContent =
staticData.arrivalTime;
```

Back-end:

For the back-end, you'll need a server to manage data from IoT sensors, perform data processing, and serve this data to the front-end. You can use Python Script.

1. Data Ingestion: Set up a system to receive data from IoT sensors. IoT devices should send location, ridership, and arrival time data to your server through APIs.

2. Data Processing: Process and clean the data as needed. This may include data validation, transformation, and storage in a database.

3. Database: Use a database like MongoDB, PostgreSQL, or MySQL to store and manage historical and real-time transit data.

4. APIs: Create APIs that allow the front-end to request real-time data from the server. These APIs should be used to retrieve data from the database and send it to the client in JSON format.

5. Real-Time Updates: Implement websockets or Server-Sent Events (SSE) to provide real-time updates to the front-end. When new data is received from the IoT sensors, push it to connected clients.

5. Data Visualization:

Data visualization is a key element in harnessing the potential of public transport optimization. Through compelling visual representations, such as interactive maps, graphs, and dashboards, transportation authorities and commuters can gain valuable insights into the performance and efficiency of public transit systems. Real-time data on vehicle locations, passenger loads, and service status can be displayed in an intuitive manner, allowing passengers to make informed travel decisions. For transit authorities, data visualization tools provide a means to track performance metrics, identify bottlenecks, and make data-driven decisions for route adjustments and service improvements.

User Interface:

Design a user-friendly interface that displays real-time transit information. The interface can include:

1. **Maps:** Display the current locations of public transport vehicles on a map using a library like Leaflet or Google Maps.

2. **Tables:** Show a list of vehicles with details such as vehicle number, current location, and estimated arrival time at key stops.

3. **Real-Time Updates**: Continuously update the information on the platform as new data from IoT sensors becomes available.

6. Alerts and Notification:

Alerts and notifications are indispensable components of public transport optimization, ensuring that both passengers and transit authorities stay informed and responsive. Commuters can receive real-time updates on service disruptions, delays, or route changes. Such notifications serve as early warning mechanisms,

mitigating potential problems and enhancing the reliability and convenience of public transport. This real-time communication not only fosters a better travel experience for passengers but also plays a vital role in optimizing transit services and improving overall urban mobility.

7. User Authentication and Security:

User authentication and security in public transport optimization is paramount for ensuring safe and efficient transit systems. Implementing robust user authentication methods, such as contactless smart cards or mobile apps with secure authentication protocols, not only helps in preventing fare evasion but also enhances passenger safety.

8. Database Management:

Database management plays a vital role in public transport optimization by efficiently storing, retrieving, and managing vast amounts of data critical for the functioning of the transit system. This includes passenger information, route schedules, vehicle maintenance records, and real-time tracking data.

A well-designed database system enables operators to make informed decisions, such as optimizing routes, improving maintenance schedules, and enhancing the overall passenger experience. Additionally, it allows for data analysis, enabling the identification of trends and areas for improvement in public transport services. Effective database management not only ensures the smooth operation of transit systems but also contributes to better service quality and cost-effectiveness, ultimately benefiting and operators alike.

Conclusion:

Public Transport Optimization in IoT leverages real-time data collection, analysis, and smart decision-making to make public transportation systems more efficient, reliable, and user-friendly while also contributing to sustainability and safety goals.