**PHASE- 3 : DEVELOPMENT PART 1**

**PUBLIC TRANSPORTATION OPTIMIZATION**

**IBM: NAAN MUDHALVAN.**

**GROUP 4 & ZONE 14**

**MENTOR: K.PRIYANKA M.E.., ASSISTANT PROFESSOR**

**TEAM MEMBERS: DOMAIN-IOT**

* + - * + **S.SUBHA DHARSHINI**
        + **A.VISHNU PRIYA**
        + **M.PRIYA**
        + **K.GAYATHRI**
        + **R.RABIKA**

**INTRODUCTION:**

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.

**0BJECTIVES:**

1. To determine the specific goals of the system, such as improving efficiency, reducing costs, enhancing passenger experience, or reducing environmental impact.

2. To develop a Smart system that could benefit RTC (Road Transport Corporation) as well as the passengers

3. To develop a business model where operator can act as an Enterprise Service Provider

4. To encourage the passenger to use public transport for commuting there by reducing traffic congestion, air pollution etc.

**IDENTIFY STAKEHOLDERS:**

Determine who will be using the system - public transportation agencies, commuters, or other stakeholders.

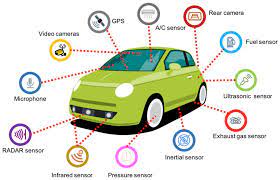
**REQUIREMENT:**

• GPS devices

• Wheel speed sensors

• Torque sensors

• Sensors measuring the health of the vehicle



**HARDWARE SELECTION:**

* Choose suitable I0T devices (sensors, GPS modules, RFID/NFC readers, etc.) to collect data from buses, trains, and stations.
* Select appropriate communication protocols (like MQTT, CoAP) and network connectivity options (Wi-Fi, cellular, LoRa) for data transmission.

**Data Collection and Sensors:**

**Install Sensors:**

Equip vehicles with sensors to collect data such as location, speed, fuel consumption, passenger count, and environmental conditions.

**Station Sensors:**

Implement sensors at stations for passenger count, occupancy, and environmental monitoring.

**Data Storage:**

Set up a secure and scalable database to store the collected data.

**RFID SENSOR**

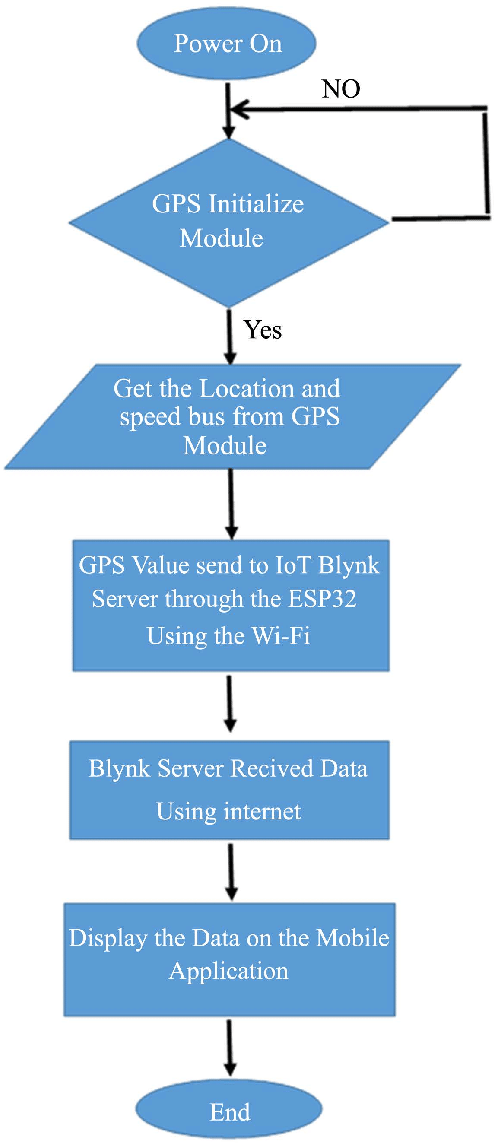
Incorporating machine learning algorithms to improve arrival time prediction accuracy based on historical data and traffic conditions is a valuable and practical use case for enhancing transportation and logistics systems. Here's a step-by-step approach to implementing such a system:

1. **Data Collection and Pre-processing:**
   * Gather historical data: Collect data on past trips, including start and end times, routes taken, traffic conditions , and any other relevant factors (weather, road closures, special events, etc.).
   * Real-time data: Integrate real-time data sources such as traffic cameras, GPS data, and weather forecasts to provide up-to-the-minute information.
   * Data pre-processing: Clean and pre-process the data, handling missing values and outliers, and converting categorical variables into numerical formats.
2. **Feature Engineering:**
   * Extract relevant features: Create features from the data that can be used for prediction, such as time of day, day of the week, road type, historical traffic congestion patterns, and more.
   * Feature selection: Use techniques like feature importance analysis or dimensionality reduction to select the most informative features.
3. **Model Selection:**
   * Choose appropriate machine learning algorithms for regression or time series forecasting. Common choices include:
     + Linear Regression
     + Random Forest Regression
     + Gradient Boosting (e.g., Boost, Light)
     + Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks for sequence data.
   * Experiment with different models and hyper parameters to find the best-performing one.
4. **Training and Validation:**
   * Split the data into training, validation, and test sets.
   * Train the machine learning model on the training data and validate its performance on the validation set.
   * Use metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE) to evaluate the model's accuracy.
5. **Hyper parameter Tuning:**
   * Fine-tune the model's hyper parameters using techniques like grid search or Bayesian optimization to optimize its performance.
6. **Real-time Prediction:**
   * Deploy the trained model to make real-time predictions for upcoming trips. Integrate it with the transportation system to provide accurate arrival time estimates to users.
7. **Continuous Monitoring and Retraining:**
   * Regularly update the model with new data to adapt to changing traffic patterns and conditions.
   * Monitor the model's performance and retrain it as needed to maintain accuracy.
8. **User Feedback Integration:**
   * Collect feedback from users about the accuracy of the predictions.
   * Use this feedback to further improve the model and address any issues or discrepancies.
9. **Scalability and Deployment:**
   * Ensure that the system can handle a high volume of prediction requests, especially during peak traffic times.
   * Consider cloud-based solutions for scalability and reliability.
10. **Privacy and Data Security:**
    * Implement robust data security and privacy measures to protect sensitive information, such as user locations and trip histories.
11. **Regulatory Compliance:**
    * Ensure that the system complies with local regulations and data protection laws, especially when dealing with user data.

By following these steps, you can create a robust machine learning-based arrival time prediction system that continually improves its accuracy and provides valuable information to users for better trip planning and transportation management.

|  |  |
| --- | --- |
|  |  |

**FLOWCHART:**



**Communication Infrastructure:**

Implement middleware for communication between IoT devices and the central server.

Ensure real-time transmission of data from vehicles and stations to the central system.

**Cloud Platform:**

Choose a reliable cloud service provider (AWS, Azure, Google Cloud) for hosting the backend infrastructure.

**Data Processing:**

Implement data processing mechanisms to analyze and make sense of the raw data collected from sensors.

**Big Data and Analytics:**

Utilize big data analytics tools to gain insights from large datasets, enabling predictive maintenance and route optimization.

**Mobile and Web Applications:**

**Passenger App:**

Develop a user-friendly mobile app for passengers to access real-time information, plan routes, and receive updates about delays or disruptions.

**Admin Dashboard:**

Create a web-based dashboard for transportation agencies to monitor the fleet, analyze data, and make informed decisions.

**Predictive Analytics and AI:**

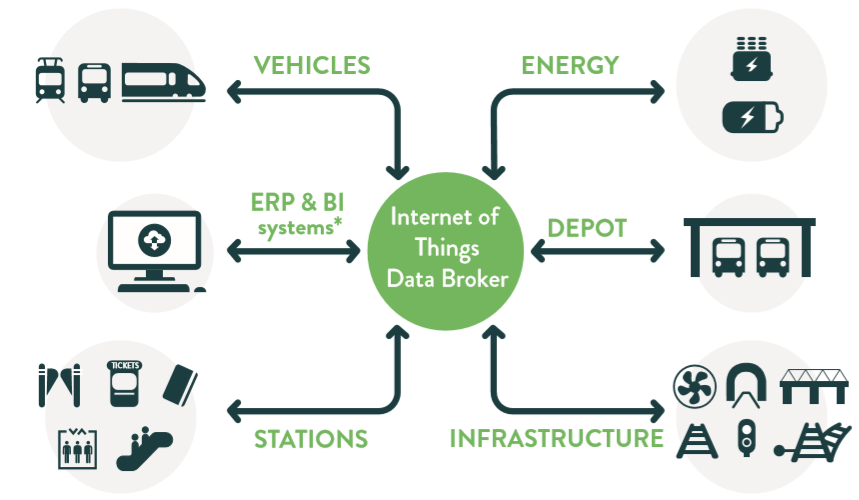
Implement predictive maintenance algorithms to foresee maintenance needs and prevent breakdowns.

Use machine learning models to predict demand, optimize routes, and enhance scheduling efficiency.

Employ IoT-enabled GPS tracking to monitor the location of public transport vehicles, ensuring efficient deployment and better coordination during peak hours.

**Energy Efficiency:**

Implement IoT sensors to monitor and control the energy consumption of public transport vehicles, reducing environmental impact and operational costs.



**Safety and Security**:

Install IoT-based surveillance systems inside vehicles and at stations to enhance passenger safety and deter criminal activities.

**Smart Bus Stops and Shelters**:

Create smart bus stops and shelters equipped with IoT sensors, offering Wi-Fi connectivity, weather updates, and charging stations for passengers.

**Demand-Responsive Services:**

Develop IoT-powered algorithms that adapt public transport routes and schedules based on real-time demand data, optimizing resources and reducing empty trips.



**Passenger Feedback Systems:**

Collect feedback from passengers through IoT-enabled kiosks or mobile apps to continuously improve public transport services based on user input.

**Noise and Pollution Monitoring:**

Use IoT sensors to measure noise levels and air quality around public transport routes, enabling authorities to take measures to reduce pollution and noise pollution.

**Accessibility Enhancements:**

Implement IoT solutions to assist passengers with disabilities, such as automatic ramps and doors, audible announcements, and real-time accessibility information.

These IoT innovations can help enhance the efficiency, safety, and convenience of public transport systems while reducing their environmental impact.

**Data Collection:**

Gather historical data on routes, including timestamps of past journeys, traffic conditions, weather, and any other relevant information that may affect travel times.

**Data Preprocessing:**

Clean and preprocess the data to handle missing values, outliers, and inconsistencies. Feature engineering can also be essential to extract relevant information from the raw data.

**Model Selection:**

Choose machine learning algorithms suitable for time series data and regression tasks. Common choices include Linear Regression, Random Forest, Gradient Boosting, and more advanced models like Long Short-Term Memory (LSTM) networks for deep learning.

**Feature Selection:**

Identify the most important features that impact arrival times and use them as inputs to your model. Feature selection techniques like feature importance scores or recursive feature elimination can help.

**Model Training:**

Split the data into training and testing sets to evaluate model performance. Train your chosen algorithm on the training data and fine-tune hyper parameters to optimize performance.

**Validation and Testing:**

Evaluate the model's accuracy and performance using the testing dataset. You can use metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) to measure prediction accuracy.

**Real-time Data Integration**:

Incorporate real-time data sources, such as traffic updates, into your model for more accurate predictions. APIs from services like Google Maps or Waze can be valuable for this purpose.

**Deployment:**

Once satisfied with the model's performance, deploy it as a service or integrate it into your navigation app. Ensure it can handle continuous updates and adapt to changing traffic conditions.

**Continuous Learning:**

Implement mechanisms for model retraining and updates to account for changing traffic patterns and new data. This ensures your predictions remain accurate over time.

**User Feedback:**

Collect user feedback to further improve your model and incorporate user suggestions or corrections into the prediction system.



**Security and Privacy:**

**Data Encryption:**

Ensure end-to-end encryption of data to maintain its confidentiality and integrity.

**Access Control:**

Implement robust access control mechanisms to restrict unauthorized access to sensitive data.

**Maintenance and Updates:**

Establish a system for regular maintenance, including software updates, security patches, and hardware checks.

Collect feedback from users and transportation staff to continuously improve the system based on real-world usage and experiences.

**Compliance and Regulations:**

Ensure that the system complies with local transportation regulations, data protection laws, and privacy standards.



**Selecting Sensors and Devices:**

GPS Sensors: Choose GPS modules that are accurate and provide real-time location data. GPS modules with integrated antennas and motion sensors are commonly used.

Passenger Counters: There are various types available, including infrared sensors, ultrasonic sensors, and weight sensors. Choose one that suits the type of public transportation vehicle and accuracy requirements.

**Power Supply:**

Ensure that the sensors are powered by the vehicle's electrical system. You might need to install voltage regulators or other power management devices to ensure stable power supply.

Implement a backup power source, like rechargeable batteries, to keep the sensors running in case of power fluctuations or vehicle shutdowns.

**Sensor Placement:**

**GPS Sensors:**

Install GPS sensors on the vehicle's roof or another location with a clear view of the sky to ensure optimal satellite signal reception.

**Passenger Counters:**

Place passenger counters strategically near vehicle entrances and exits, ensuring they cover all boarding and alighting points.



**Data Transmission:**

**Communication Protocols:**

Choose suitable communication protocols such as cellular networks (3G/4G/5G), Wi-Fi, or LoRaWAN for transmitting data from sensors to the central server.

**Data Security:**

Implement encryption and other security measures to protect the data transmitted from sensors to the server.

**Data Processing and Storage:**

**Edge Computing:**

Consider implementing edge computing to process data locally within the vehicle. This can reduce latency and bandwidth usage by sending only relevant data to the central server.

**Cloud Storage:**

Store the collected data securely in a cloud-based database. Cloud platforms like AWS, Azure, or Google Cloud offer scalable and reliable storage solutions.

**Real-time Monitoring and Analytics:**

Develop a real-time monitoring dashboard for transportation operators to track vehicle locations, passenger counts, and other relevant data.

Implement analytics tools to process the data and generate insights, helping in route optimization, scheduling, and demand forecasting.

**Calibration:**

Calibrate passenger counters regularly to maintain accuracy, especially if they rely on technologies like infrared or ultrasonic sensors.

Stay updated with advancements in sensor technologies. Plan for future sensor upgrades or additions to enhance data accuracy and gather more detailed information.

To send real-time location and ridership data from IoT sensors to a transit information platform, you can create a Python script that collects data from sensors and sends it to the platform using HTTP requests or MQTT (Message Queuing Telemetry Transport protocols.

**PROGRAM:**

import time

import random

class Bus:

def \_\_init\_\_(self, bus\_id, route):

self.bus\_id = bus\_id

self.route = route

self.current\_location = None

defupdate\_location(self):

# Simulate IoT device data

latitude = random.uniform(35.0, 36.0)

longitude = random.uniform(-90.0, -89.0)

self.current\_location = (latitude, longitude)

defget\_location(self):

return self.current\_location

class Passenger:

def \_\_init\_\_(self, passenger\_id):

self.passenger\_id = passenger\_id

class SmartTransportationSystem:

def \_\_init\_\_(self):

self.buses = []

self.passengers = []

defadd\_bus(self, bus\_id, route):

bus = Bus(bus\_id, route)

self.buses.append(bus)

defadd\_passenger(self, passenger\_id):

passenger = Passenger(passenger\_id)

self.passengers.append(passenger)

deftrack\_buses(self):

while True:

for bus in self.buses:

bus.update\_location()

print(f"Bus {bus.bus\_id} - Location: {bus.get\_location()}")

time.sleep(10) # Simulate updates every 10 seconds

if \_\_name\_\_ == "\_\_main\_\_":

transportation\_system = SmartTransportationSystem()

# Add buses and passengers

transportation\_system.add\_bus(1, "Route A")

transportation\_system.add\_bus(2, "Route B")

transportation\_system.add\_passenger(1)

transportation\_system.add\_passenger(2)

# Start tracking buses (simulated IoT updates)

transportation\_system.track\_buses()

**CONCLUSION:**

The integration of IOT technology into smart public transportation has the potential to revolutionize the way public transportation is managed, optimizing transportation routes, improving the overall passenger experience, and reducing emissions. As IOT technology continues to evolve, we can expect to see even more innovative applications means public transportation, further enhancing the efficiency and effectiveness of public transportation system.