PUBLIC TRANSPORT OPTIMIZATION USING IOT

Project Planning:

Before diving into coding, need to plan project thoroughly. Identifying our goals, target audience, and the specific features want to implement. Decide on the IoT sensors use to gather real-time data.

Selecting IoT Sensors:

To collect real-time location, ridership, and arrival time data, we will need IoT sensors. These sensors may include GPS devices for location data, passenger counters for ridership data, and RFID/NFC sensors for arrival time tracking.

Database Setup:

Creating a database to store the data from IoT sensors.can use technologies like MySQL, PostgreSQL, or NoSQL databases like MongoDB, depending on our data structure and requirements.

Front-End Development:

Building the user interface of your platform using HTML, CSS, and JavaScript. Here's what the front-end could include:

Dashboard: A central dashboard to display real-time transit information.
Interactive Map: Use libraries like Leaflet or Google Maps to show real-time bus/tram/train locations.
Graphs and Charts: Display ridership data in the form of charts or graphs.
Arrival Time Predictions: Calculate and display predicted arrival times based on real-time data.
User Authentication: Implement user accounts for customization and personalization.
Responsive Design: Ensure that your platform works on various devices and screen sizes.

Back-End Development:

Developing the server-side of your platform using a programming language like Node.js, Python, or Ruby. Key back-end tasks include:

API Integration: Building APIs to receive data from IoT sensors and provide it to the front-end.
Data Processing: Processing and clean the data from sensors to make it suitable for display.
Real-Time Data Updates: Using WebSockets or Server-Sent Events (SSE) to push real-time updates to the front-end.
User Management: Handling user accounts, authentication, and authorization.
Database Integration: Connecing our back-end to the database where we store sensor data.

IoT Sensor Integration:

Set up our IoT sensors to send data to your back-end via APIs or other communication protocols. Ensuring that the data sent is in a format that your system can process.

Real-Time Transit Data Display:

Implementing real-time data display on our platform. This can include live maps, arrival time predictions, ridership statistics, and other relevant information.

Testing:

Thoroughly test your platform to ensure that it works correctly. Test different scenarios, user interactions, and real-time data updates.

CODE FOR FRONT END

Html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Public Transit Information</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<header>
```

```
<h1>Real-Time Transit Information</h1>
</header>
<main>
<div id="map-container">
<!-- Real-time map will be inserted here -->
</div>
<div id="ridership">
<h2>Ridership Data</h2>
<div id="ridership-chart">
<!-- Ridership chart will be inserted here -->
</div>
</div>
<div id="arrival-times">
<h2>Arrival Times</h2>
<ul id="arrival-times-list">
<!-- Arrival time data will be inserted here -->
</ul>
</div>
</main>
<script src="app.js"></script>
</body>
</html>
```

CSS

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

header {
background-color: #0073e6;
color: #fff;
text-align: center;
```

```
padding: 20px;
}

main {
max-width: 1200px;
margin: 0 auto;
padding: 20px;
}

#map-container {
height: 400px;
}

#ridership {
margin-top: 20px;
padding: 20px;
background-color: #f2f2f2;
}

#arrival-times {
margin-top: 20px;
}

#ridership-chart {
height: 300px;
  }
```

JS

```
.//  Fetch and display real-time map data using Leaflet
const map = L.map('map-container').setView([latitude, longitude], zoomLevel);
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png').addTo(map);

#Fetch and display ridership chart using a charting library like Chart.js
const ridershipData = {
```

```
    labels: ['January', 'February', 'March', 'April', 'May'],
    datasets: [
        {
            label: 'Ridership',
            data: [1200, 1500, 1300, 1800, 1600],
            borderColor: 'blue',
            backgroundColor: 'rgba(0, 115, 230, 0.2)',
        },
    ],
};

const ctx = document.getElementById('ridership-chart').getContext('2d');
new Chart(ctx, {
    type: 'line',
    data: ridershipData,
});

// Fetch and display arrival time data
const arrivalTimes = ['09:15 AM', '09:30 AM', '09:45 AM', '10:00 AM', '10:15 AM'];
const arrivalTimesList = document.getElementById('arrival-times-list');

arrivalTimes.forEach((time) => {
    const li = document.createElement('li');
    li.textContent = time;
    arrivalTimesList.appendChild(li);
});
```