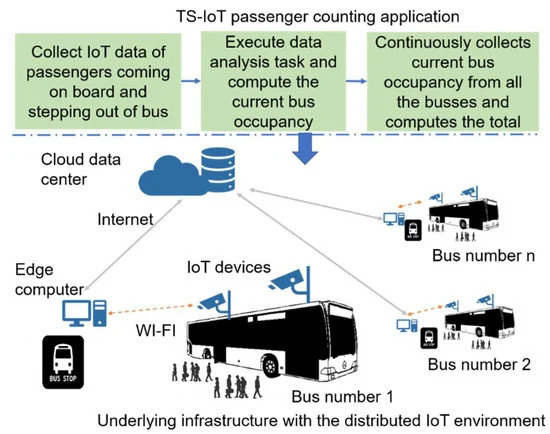
**(R.SANTHOSH – AU732721106031)**

**PUBLIC TRANSPORTATION OPTIMIZATION**



**User Interface Design:** Design a user-friendly interface that can be accessed via web browsers on desktop and mobile devices. Consider the following elements:

**Map:** Display a map showing the real-time location of transit vehicles.

**Data Visualization:** Create charts or graphs to show ridership statistics.

**Real-time Updates:** Implement a dashboard with the latest information.

**Search Functionality:** Allow users to search for specific routes or stops.

**HTML Structure:** Develop the HTML structure for the web page. Here’s a basic example:

**PROGRAM:-**

<!DOCTYPE html>

<html>

<head>

<title>Real-time Transit Information</title>

<link rel=”stylesheet” type=”text/css” href=”style.css”>

</head>

<body>

<div id=”map”></div>

<div id=”data-visualization”></div>

<div id=”real-time-updates”></div>

<div id=”search-bar”>

<input type=”text” id=”search-input” placeholder=”Search for routes or stops”>

</div>

<script src=”script.js”></script>

</body>

</html>

**CSS Styling:** Style the elements using CSS to make the interface visually appealing and responsive.

**JavaScript Front-end Logic:**

Use JavaScript to interact with the back-end and fetch real-time data.

Display real-time vehicle location on the map using libraries like Leaflet or Google Maps.

Create charts or graphs for ridership data using libraries like Chart.js or D3.js.

Update real-time information on the dashboard.

Implement search functionality to filter and display specific routes or stops.

**Back-end Development:**

**IoT Sensors:** Set up IoT sensors on transit vehicles to collect data, including GPS location, ridership counts, and estimated arrival times. IoT devices should transmit this data to your back-end system.

**Data Processing:** Develop a back-end server to receive, process, and store the data from IoT sensors. You can use technologies like Node.js, Python (with frameworks like Flask or Django), or any other language you are comfortable with.

**APIs:** Create APIs to serve real-time data to the front-end. For instance, you can have endpoints for vehicle location, ridership statistics, and arrival time predictions.

**Database:** Store the historical and real-time data in a database (e.g., PostgreSQL, MongoDB) for analysis and retrieval.

**Real-time Updates:** Implement mechanisms for sending real-time updates to the front-end, such as WebSockets or server-sent events (SSE).

**Authentication and Security:** Ensure data security by implementing authentication and authorization mechanisms, especially if sensitive data is involved.

**Integration:**

Connect the front-end and back-end by making API requests from the front-end to fetch real-time data.

Set up automated processes to update real-time data and send alerts when there are significant delays or issues.

Implement error handling and graceful degradation for the platform to ensure it remains functional even if certain sensors or components fail.

**Testing and Deployment:**

Test the platform rigorously to ensure it works as expected.

Deploy the front-end and back-end to a web server or cloud platform.

Monitor the platform’s performance and ensure it can handle the expected user load.

Regularly update and maintain the system to accommodate changes in transit data or user requirements.

Remember to consult relevant data privacy and regulatory requirements when handling transit data and IoT devices. This project can be complex, so it’s important to plan and implement it step by step, and possibly collaborate with a team with expertise in web development, IoT, and data management.

**Team Members:-**

1. **K.S. HARI PRASATH**
2. **M. HARISH**
3. **K. KISHORE SARATHI**
4. **R. SANTHOSH**