PUBLIC TRANSPORT OPTIMIZATION

1. INTRODUCTION:

Project Overview:

The Real-Time Transit Information Platform is a web-based application designed to display simulated real-time transit data for a fictional transit station. This documentation provides an overview of the project, its objectives, and the technologies used.

Objectives:

The main objectives of the project are as follows

Simulate real-time transit data updates for a fictional transit station.

Demonstrate the use of web development technologies (HTML, CSS, JavaScript) to create a dynamic platform.

2. HTML STRUCTURE (INDEX.HTML):

The index.html file serves as the entry point for the application.

Code Explanation:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Real-Time Transit Information</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Real-Time Transit Information</h1>
    </header>
    <section id="data-display">
        <p>Loading real-time data...</p>
    </section>
    <script src="script.js"></script>
</body>
</html>
```

3. JAVASCRIPT LOGIC (SCRIPT.JS)

The script.js file contains JavaScript code that simulates real-time data updates.

Code Explanation:

```
const dataDisplay = document.getElementById('data-display');
```

```
function updateData() {
    const location = 'Station A';
    const ridership = Math.floor(Math.random() * 100);
    const arrivalTime = (Math.random() * 30).toFixed(1);

    const data = `
        <p>Location: ${location}</p>
        <p>Ridership: ${ridership} passengers</p>
        <p>Estimated Arrival Time: ${arrivalTime} minutes</p>
    `;

    dataDisplay.innerHTML = data;
}

// Update data every 10 seconds (simulating real-time updates)
updateData();
setInterval(updateData, 10000);
```

4. CSS STYLING (STYLES.CSS):

The styles.css file provides styling for the webpage.

Code Explanation:

```
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    margin: 0;
    padding: 0;
}

header {
    background-color: #0074D9;
    color: #fff;
    text-align: center;
    padding: 10px;
}

h1 {
    margin: 0;
}

#data-display {
    background-color: #fff;
    padding: 20px;
    margin: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
}
```

5. PROJECT ARCHITECTURE

High-Level Overview

The project consists of a simple webpage with HTML, CSS, and JavaScript for the frontend. It simulates real-time data updates without the use of actual IoT sensors. The data is generated and displayed on the page.

Data Flow

Data is generated and updated in real time using JavaScript. The frontend receives data updates and displays them in the designated section.

6. REAL-TIME DATA SIMULATION

Data Source

The project simulates data for a fictional transit station. Data sources are not real IoT sensors but generated within the JavaScript code.

Data Generation Logic

Data is generated with random values for location, ridership, and estimated arrival time.

Real-Time Updates

The JavaScript code simulates real-time updates by refreshing data every 10 seconds.

7. USER INTERFACE:

The user interface includes a header and a data display section, styled using CSS.

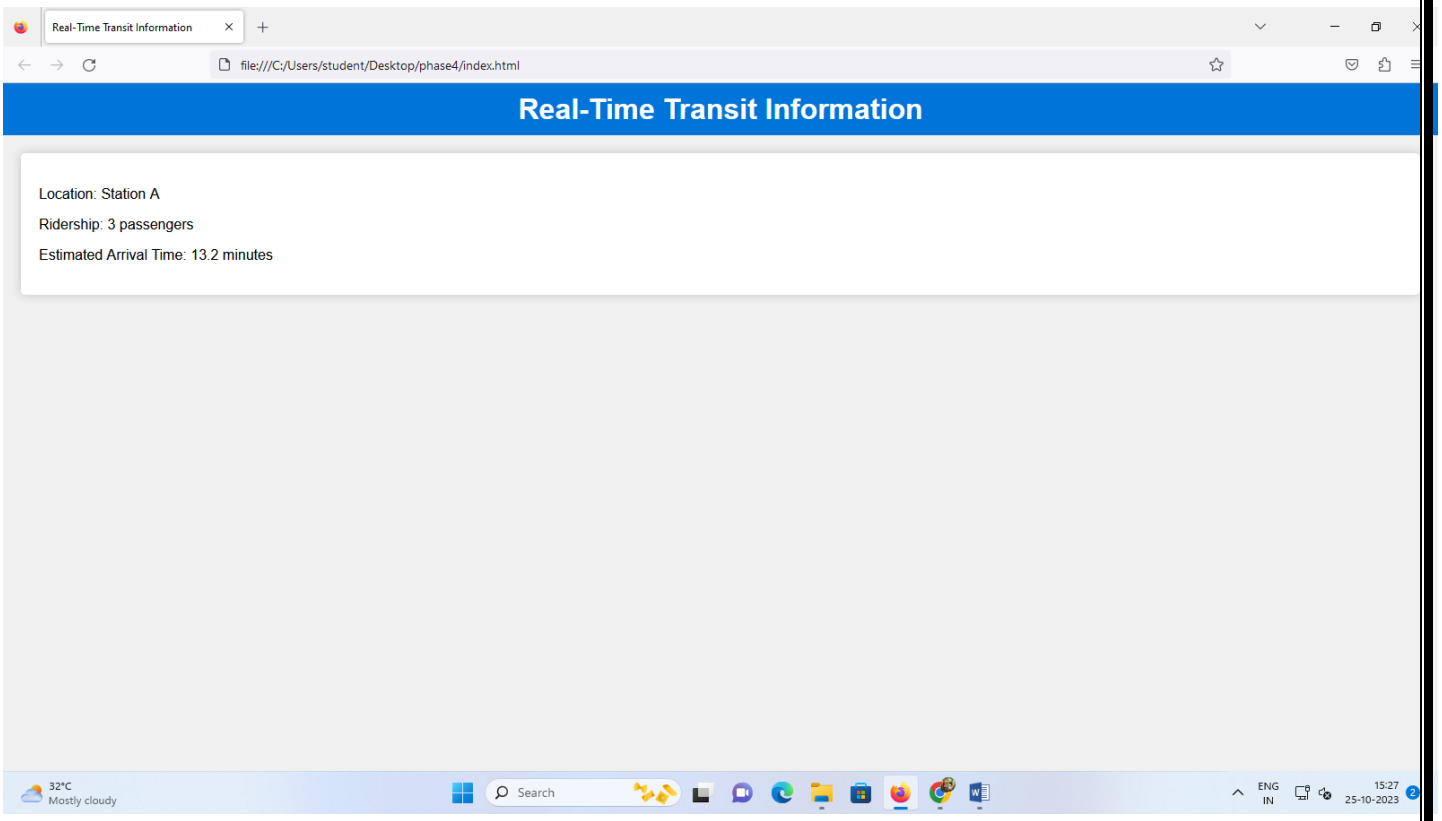
8. TESTING:

The project can be tested by opening the index.html file in a web browser. Real-time data updates are simulated, and you can observe the changes.

9. DEPLOYMENT:

The project is deployed as a static website, and it can be hosted on any web server or platform capable of serving HTML, CSS, and JavaScript files.

OUTPUT:



10. CONCLUSION:

The Real-Time Transit Information Platform is a simple example of using web development technologies to simulate and display real-time data updates. It serves as a basic educational resource for understanding how real-time data can be presented in a web application.