**PHASE- 4 : DEVELOPMENT PART 2**

**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:**

Optimizing public transport involves various aspects such as route planning, scheduling, and passenger information systems. It can be a complex task depending on the specific goals and constraints. Below, I'll provide a basic web development technologies example for optimizing transportation. This is a simplified illustration, and real-world public transport optimization projects are typically much more complex.

Creating a real-time transit information platform involves several steps. Here's a high-level outline of what you should consider:

**Define Requirements:**

Clearly define the requirements for your platform. What kind of transit information do you want to display? What IoT sensors will you be using, and how will they transmit data?

**Select IoT Sensors:**

Choose appropriate IoT sensors for collecting real-time data. These could include GPS sensors for location, passenger counters for ridership, and sensors on vehicles for arrival time information.

**Data Collection and Transmission:**

Set up a system to collect data from the IoT sensors. This often involves using microcontrollers or IoT platforms to transmit data to a central server.

**Database:**

Create a database to store the real-time data. You may use technologies like MySQL, MongoDB, or others, depending on your specific needs.

**Web Development:**

Build the web platform using HTML, CSS, and JavaScript. You can use frameworks like React, Angular, or Vue.js for a more dynamic interface.

**Real-Time Updates:**

Implement mechanisms like WebSockets or Server-Sent Events to enable real-time updates on your platform.

**Data Visualization:**

Use libraries like D3.js or Chart.js to create visually appealing displays for location, ridership, and arrival time data.

**User Interface:**

Design an intuitive user interface that allows users to interact with the data easily. Consider mobile responsiveness for users on different devices.

**Security:**

Ensure data security and user privacy by implementing authentication and authorization mechanisms.

**Testing:**

Rigorously test the platform to identify and fix any bugs or performance issues.

**Deployment:**

Deploy your platform to a web server. Consider using cloud platforms like AWS, Azure, or Heroku for scalability.

**Maintenance:**

Continuously monitor and maintain the platform to ensure it remains functional and up-to-date.

**Documentation:**

Create user and developer documentation to guide users and future developers working on the project.

Remember to adapt the above steps to your specific project's needs and size. Developing a real-time transit information platform is a complex task, so consider the scale and resources available for your project.

Here's a basic example of a web development technologies(i.e., html , css javascript)

Html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="styles.css">

<title>Smart Public Transportation</title>

</head>

<body>

<div class="container">

<h1>Smart Public Transportation System</h1>

<label for="routes">Select Route:</label>

<select id="routes">

<option value="Route1">Route 1</option>

<option value="Route2">Route 2</option>

</select><br>

<label for="busStops">Select Bus Stop:</label>

<select id="busStops">

<option value="StopA">Stop A</option>

<option value="StopB">Stop B</option>

<option value="StopC">Stop C</option>

<option value="StopD">Stop D</option>

</select><br>

<button onclick="findNextBus()">Find Next Bus</button>

<p id="result"></p>

</div>

<script src="script.js"></script>

</body>

</html>

Css:

body {

font-family: Arial, sans-serif;

}

.container {

max-width: 300px;

margin: 0 auto;

text-align: center;

padding: 20px;

border: 1px solid #ccc;

border-radius: 10px;

margin-top: 50px;

}

button {

margin-top: 10px;

cursor: pointer;

}

Javascript:

function findNextBus() {

const selectedRoute = document.getElementById("routes").value;

const selectedStop = document.getElementById("busStops").value;

const routes = {

"Route1": ["StopA", "StopB", "StopC", "StopD"],

"Route2": ["StopD", "StopC", "StopB", "StopA"]

};

const stops = routes[selectedRoute];

const currentIndex = stops.indexOf(selectedStop);

const nextStopIndex = (currentIndex + 1) % stops.length;

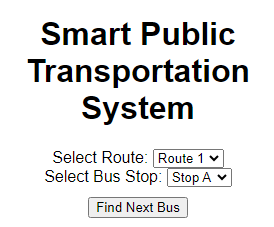
const nextStop = stops[nextStopIndex];

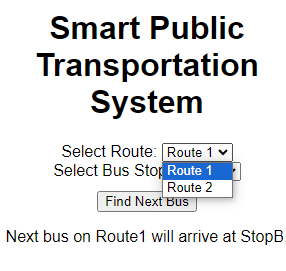
const result = `Next bus on ${selectedRoute} will arrive at ${nextStop}.`;

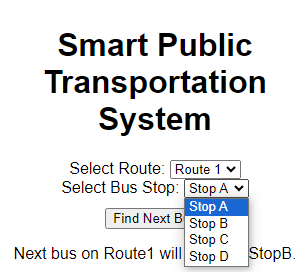
document.getElementById("result").textContent = result;

}

**Output:**

****

****



Designing a public transport optimization platform to receive and display real-time location, ridership, and arrival time data from IoT sensors involves several components. Here's a high-level architectural overview of how such a system could be designed:

**IoT Sensors:**

Deploy IoT sensors on public transport vehicles (buses, trams, trains) to collect data. These sensors should capture:

**Real-time location**: GPS data to track the vehicle's position.

**Ridership data:** Passenger counting sensors, such as weight sensors or infrared sensors.

**Arrival time data**: Sensors at stops to detect when a vehicle arrives or departs.

**Data Collection and Processing:**

Set up a centralized data collection system to receive data from IoT sensors in real-time.

Process and filter the incoming data to extract relevant information, such as vehicle ID, location, passenger counts, and timestamps.

**Data Storage:**

Store the processed data in a database system. You might use a database management system like PostgreSQL or a NoSQL database, depending on the scale and complexity of your project.

**API and Backend:**

**Develop a backend system with APIs to:**

Receive data from the IoT sensors and update the database.

Provide access to real-time data for external services and the front-end.

**Front-End Interface:**

Create a user-friendly web interface using HTML, CSS, and JavaScript to display the real-time information to users.

Use JavaScript libraries for mapping and data visualization (e.g., Leaflet, D3.js).

**Real-Time Data Display:**

On the front end, continuously request and display data from the backend API to show real-time vehicle locations on a map.

Display current ridership data and vehicle arrival times for specific stops.

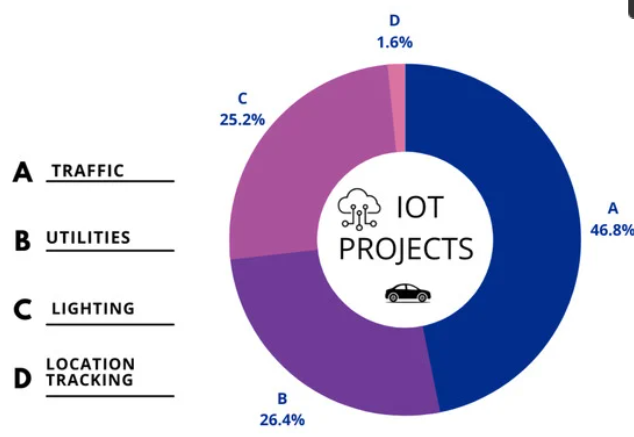
**Route Planning and Optimization:**

Implement algorithms for route planning and optimization, using the real-time data to recommend optimal routes and transit options.

**User Features:**

Allow users to search for routes, set preferences, and receive alerts for delays or changes.

Display historical data and trends in ridership and on-time performance.



Here's a basic example of a web page that allows users to select their starting and ending points for a public transport route:

html

<!DOCTYPE html>

<html>

<head>

<title>Public Transport Optimization</title>

<style>

/\* Add your CSS styles here \*/

body {

font-family: Arial, sans-serif;

}

</style>

</head>

<body>

<h1>Public Transport Optimization</h1>

<form>

<label for="start">Starting Point:</label>

<input type="text" id="start" placeholder="Enter starting point">

<label for="end">Ending Point:</label>

<input type="text" id="end" placeholder="Enter ending point">

<button onclick="optimizeRoute()">Optimize Route</button>

</form>

<div id="result">

<!-- Display optimization results here -->

</div>

<script>

function optimizeRoute() {

// Add your JavaScript logic for route optimization here

// You would need to make API calls, implement algorithms, and display the optimized route.

// This is a simplified example, and the actual logic can be quite complex.

document.getElementById('result').innerHTML = 'Optimized route goes here';

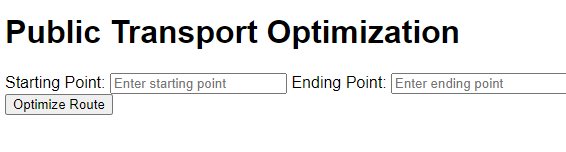
}

</script>

</body>

</html>

**Output:**

****

**EXAMPLE:**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Public Transport Optimization</title>**

**</head>**

**<body>**

**<h1>Public Transport Optimization Example</h1>**

**<h2>Bus Routes</h2>**

**<ul>**

**<li>Route 1: Stops - A, B, C, D</li>**

**<li>Route 2: Stops - D, E, F, G</li>**

**<!-- Add more routes and stops as needed -->**

**</ul>**

**<h2>Train Routes</h2>**

**<ul>**

**<li>Line A: Stations - Station 1, Station 2, Station 3</li>**

**<li>Line B: Stations - Station 3, Station 4, Station 5</li>**

**<!-- Add more train lines and stations as needed -->**

**</ul>**

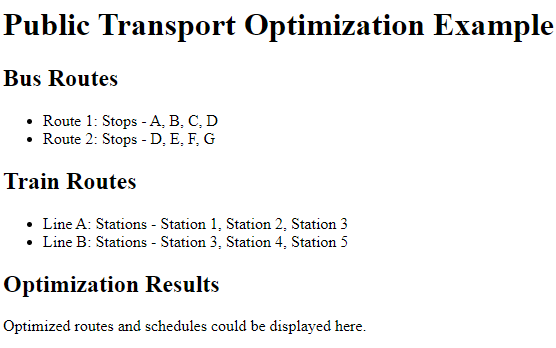
**<h2>Optimization Results</h2>**

**<p>Optimized routes and schedules could be displayed here.</p>**

**<!-- You can add JavaScript for interactivity and further optimization -->**

**</body>**

**</html>**

**Output: **

**Python code:**

Import random  
  
# Constants  
POPULATION\_SIZE = 10  
MUTATION\_RATE = 0.1  
GENERATIONS = 100  
  
# Sample data (can be replaced with real data)  
bus\_stops = [(1, 1), (2, 2), (3, 3), (4, 4), (5, 5)]  
bus\_routes = [[0, 1, 2, 3, 4], [4, 3, 2, 1, 0]]  # Sample initial routes  
  
  
# Function to calculate the total travel time for a bus route  
def calculate\_total\_time(route):  
    total\_time = 0  
    for i in range(len(route) - 1):  
        total\_time += abs(bus\_stops[route[i]][0] - bus\_stops[route[i + 1]][0]) + abs(  
            bus\_stops[route[i]][1] - bus\_stops[route[i + 1]][1]  
        )  
    return total\_time  
  
  
# Genetic Algorithm  
def genetic\_algorithm():  
    population = [bus\_routes] \* POPULATION\_SIZE  
  
    for generation in range(GENERATIONS):  
        population = sorted(population, key=lambda x: calculate\_total\_time(x))  
        fittest = population[0]  
  
        new\_population = [fittest]  
  
        for \_ in range(POPULATION\_SIZE - 1):  
            parent1 = random.choice(population[: POPULATION\_SIZE // 2])  
            parent2 = random.choice(population[: POPULATION\_SIZE // 2])  
            child = []  
            for i in range(len(bus\_stops)):  
                if random.random() < MUTATION\_RATE:  
                    child.append(random.randint(0, len(bus\_stops) - 1))  
                else:  
                    if random.random() < 0.5:  
                        child.append(parent1[i])  
                    else:  
                        child.append(parent2[i])  
            new\_population.append(child)  
  
        population = new\_population  
  
    return sorted(population, key=lambda x: calculate\_total\_time(x))[0]  
  
  
# Run the Genetic Algorithm and print optimized routes  
optimized\_routes = genetic\_algorithm()  
for i, route in enumerate(optimized\_routes):  
    print(f"Bus {i + 1} Route: {route}")

long get\_distance1() {

  // Start a new measurement:

  digitalWrite(PIN\_TRIG1, HIGH);

  delayMicroseconds(10);

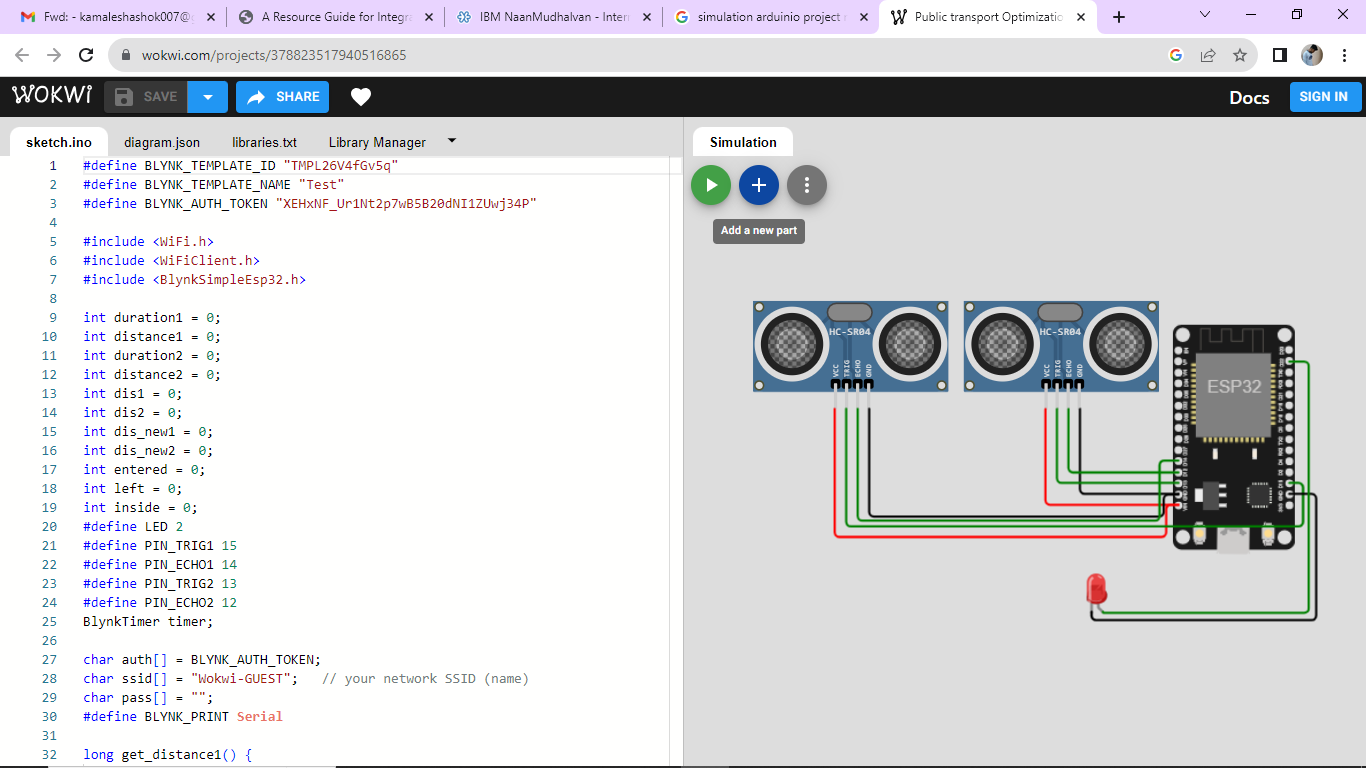
  digitalWrite(PIN\_TRIG1, LOW);

  // Read the result:

  duration1 = pulseIn(PIN\_ECHO1, HIGH);

  distance1 = duration1 / 58;

  return distance

}

Conclusion:

The Internet of Things technology will continue to [improve the passenger experience](https://internetofthingsagenda.techtarget.com/blog/IoT-Agenda/How-IoT-will-improve-public-transportation) for public transportation by offering real-time vehicle tracking, improved responses in the event of an unexpected event, and personalized travel information. As cities become more congested and as more people look for ways to go green, public transit will become a very attractive option for people looking to forgo using their personal vehicles. IoT technology will only improve public transit and as public transit improves, there will be converts left and right.