IOT PHASE 4

PUBLIC TRANSPORT OPTIMIZATION

AIM:

Creating a full-fledged mobile app for transport optimization which is a complex task that requires extensive development and resources. The aim of creating an app for traffic updates and route recommendations is to provide users with real-time information about traffic conditions and suggest the best routes based on current traffic data. This helps users to make informed decisions and optimize their travel time.

ALGORITHM:

1.Collect the traffic data from the user using API

2.Analyse the passenger load and route data to allocate the right number of buses to each route.

3.Create bus schedules that align with demand, ensuring timely arrivals and departures and adjust schedules based on real-time data, such as traffic delays.

4. Use algorithms like Dijkstra's or A* to find the shortest or fastest routes between stops, considering traffic conditions.

5. Use dynamic pricing algorithms to adjust fares based on demand, time of day, or other factors.

6. Implement algorithms to provide buses with priority at traffic signals, reducing delays.

COMPONENTS REQUIRED:

HTML components

1. **Page Structure:**

HTML provides the structure of your web pages, including headings, paragraphs, lists, and tables to organize content.

2. Forms:

Create forms to collect user input, such as search queries, location information, and feedback.

3. Interactive Elements:

Implement interactive elements like buttons, links, and checkboxes for user interaction.

4. Maps:

Embed maps using HTML components to display bus routes, stops, and real-time bus locations.

5. Tables and Lists:

Use tables and lists to present data, such as schedules, routes, and fare information.

6. Media Elements:

Include images and videos to enhance user experience and provide visual information.

7. Geolocation API:

Utilize the HTML5 Geolocation API to access the user's location for route planning and real-time bus tracking.

JAVASCRIPT components

1. Event Handling:

JavaScript allows you to define event handlers for user actions, such as button clicks or form submissions.

2. Asynchronous Operations:

Use JavaScript's asynchronous capabilities to fetch data from APIs and databases without blocking the user interface.

3. API Integration:

Interact with APIs to retrieve and display real-time information, such as bus locations, arrival times, and route data.

4. Data Visualization:

Utilize JavaScript libraries and frameworks, such as D3.js or Chart.js, to create data visualizations like graphs or charts for route optimization or passenger demand analysis.

5.Real-Time Updates:

Use JavaScript's timer functions to periodically fetch and update real-time data, such as bus locations and estimated arrival times.

6. User Feedback:

Create feedback forms and mechanisms to allow users to submit comments, ratings, or suggestions for system improvement.

7. Local Storage:

Store user preferences, settings, or frequently accessed data on the client-side using JavaScript's local storage.

8. Error Handling:

Use JavaScript to handle errors gracefully and provide meaningful error messages to users when issues occur.

PROGRAM:

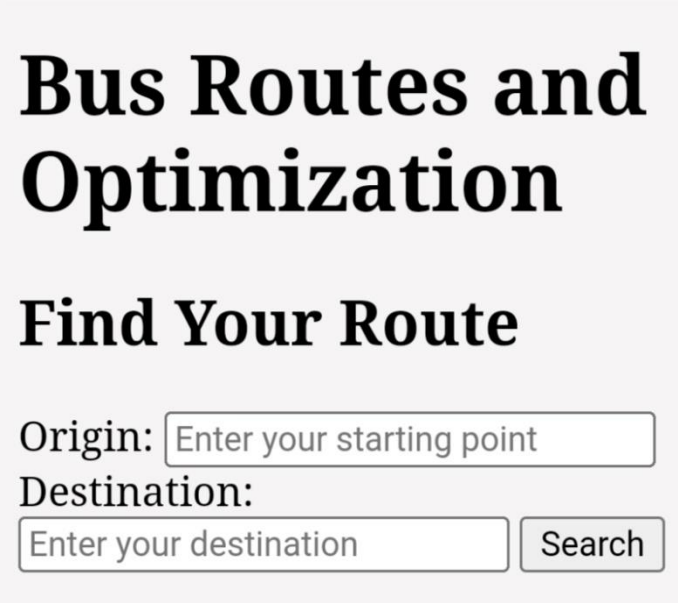
USING HTML

```
<!DOCTYPE html>
<html>
<head>
   <title>Public Transport Optimization</title>
```

```
    <!--Include external CSS files for styling -->
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <header>
        <h1>Bus Routes and Optimization</h1>
    </header>
    <main>
        <section id="search">
            <h2>Find Your Route</h2>
            <form id="route-search-form">
                <label for="origin">Origin:</label>
                <input type="text" id="origin" name="origin" placeholder="Enter your starting point"
required>

                <label for="destination">Destination:</label>
                <input type="text" id="destination" name="destination" placeholder="Enter your
destination" required>

                <button type="submit">Search</button>
            </form>
        </section>

        <section id="results">
            <h2>Search Results</h2>
            <div id="route-list">
                <!-- Route results will be displayed here dynamically using JavaScript -->
            </div>
        </section>
    </main>

    <footer>
        <p>&copy; 2023 Public Transport Optimization</p>
    </footer>
```

```
    <!--Include external JavaScript files for interactivity -->
    <script src="app.js"></script>
</body>
</html>
```

OUTPUT:

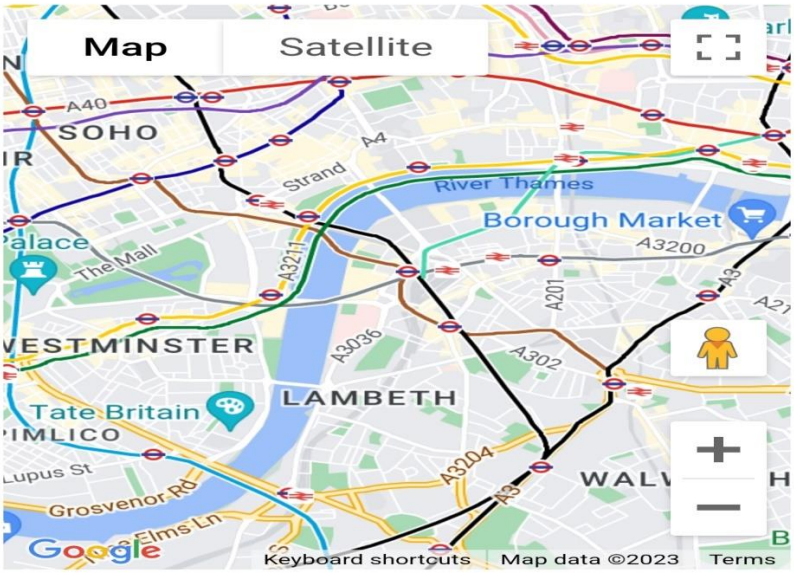


USING JAVASCRIPT

```
function initMap() {
  const bounds = new google.maps.LatLngBounds();
  const markersArray = [];
  const map = new google.maps.Map(document.getElementById("map"), {
    center: { lat: 55.53, lng: 9.4 },
    zoom: 10,
  });
  // initialize services
  const geocoder = new google.maps.Geocoder();
  const service = new google.maps.DistanceMatrixService();
  // build request
  const origin1 = { lat: 55.93, lng: -3.118 };
  const origin2 = "Greenwich, England";
  const destinationA = "Stockholm, Sweden";
  const destinationB = { lat: 50.087, lng: 14.421 };
  const request = {
    origins: [origin1, origin2],
    destinations: [destinationA, destinationB],
    travelMode: google.maps.TravelMode.DRIVING,
    unitSystem: google.maps.UnitSystem.METRIC,
    avoidHighways: false,
    avoidTolls: false,
  };

  // put request on page
  document.getElementById("request").innerText = JSON.stringify(
    request,
    null,
```

```
    2,
  );
  // get distance matrix response
  service.getDistanceMatrix(request).then((response) => {
    // put response
    document.getElementById("response").innerText = JSON.stringify(
      response,
      null,
      2,
    );

    // show on map
    const originList = response.originAddresses;
    const destinationList = response.destinationAddresses;

    deleteMarkers(markersArray);

    const showGeocodedAddressOnMap = (asDestination) => {
      const handler = ({ results }) => {
        map.fitBounds(bounds.extend(results[0].geometry.location));
        markersArray.push(
          new google.maps.Marker({
            map,
            position: results[0].geometry.location,
            label: asDestination ? "D" : "O",
          }),
        );
      };
      return handler;
    };

    for (let i = 0; i < originList.length; i++) {
      const results = response.rows[i].elements;

      geocoder
        .geocode({ address: originList[i] })
        .then(showGeocodedAddressOnMap(false));

      for (let j = 0; j < results.length; j++) {
        geocoder
          .geocode({ address: destinationList[j] })
          .then(showGeocodedAddressOnMap(true));
      }
    }
  });
}

function deleteMarkers(markersArray) {
  for (let i = 0; i < markersArray.length; i++) {
    markersArray[i].setMap(null);
  }

  markersArray = [];
}
```

OUTPUT:



DEVELOPMENT:

1. Data Collection and Analysis:

Gather data on existing passenger demand, travel patterns, and congestion points.

Analyze historical data to identify popular routes, high-traffic areas, and peak travel times.

2. Define Objectives:

Determine the goals of your bus route development, such as reducing travel time, increasing coverage, or improving connectivity.

3. Network Design:

Create a network of potential bus routes based on the collected data and objectives.

4. Route Modeling:

Utilize transportation modeling software to design and simulate different route options, taking into account factors like traffic conditions and stop locations.

5. Demand Forecasting:

Use statistical models or machine learning algorithms to forecast passenger demand on different routes and at various times of the day.

6. Stop Location Selection:

Identify and evaluate potential bus stop locations, considering factors like proximity to key destinations, pedestrian accessibility, and passenger demand.

7. Frequency and Schedule Planning:

Determine the optimal frequency and schedules for each route to meet passenger demand while avoiding over- or under-servicing.

8. Connection Points:

Plan transfer points where different bus routes intersect or connect with other modes of transport (e.g., trains, trams) to provide a seamless transit experience.

9. Service Reliability:

Ensure that bus routes are designed to minimize delays and disruptions, taking into account traffic congestion, road conditions, and planned maintenance.

10. Environmental Considerations:

Optimize routes to reduce emissions and fuel consumption, which may include avoiding congested areas or implementing eco-friendly buses.

11. Community Engagement:

Seek input from the local community and stakeholders to understand their needs and concerns and make adjustments accordingly.

12. Safety Measures:

Prioritize safety in route design, considering factors such as pedestrian crossings, school zones, and accident-prone areas.

13. Real-Time Data Integration:

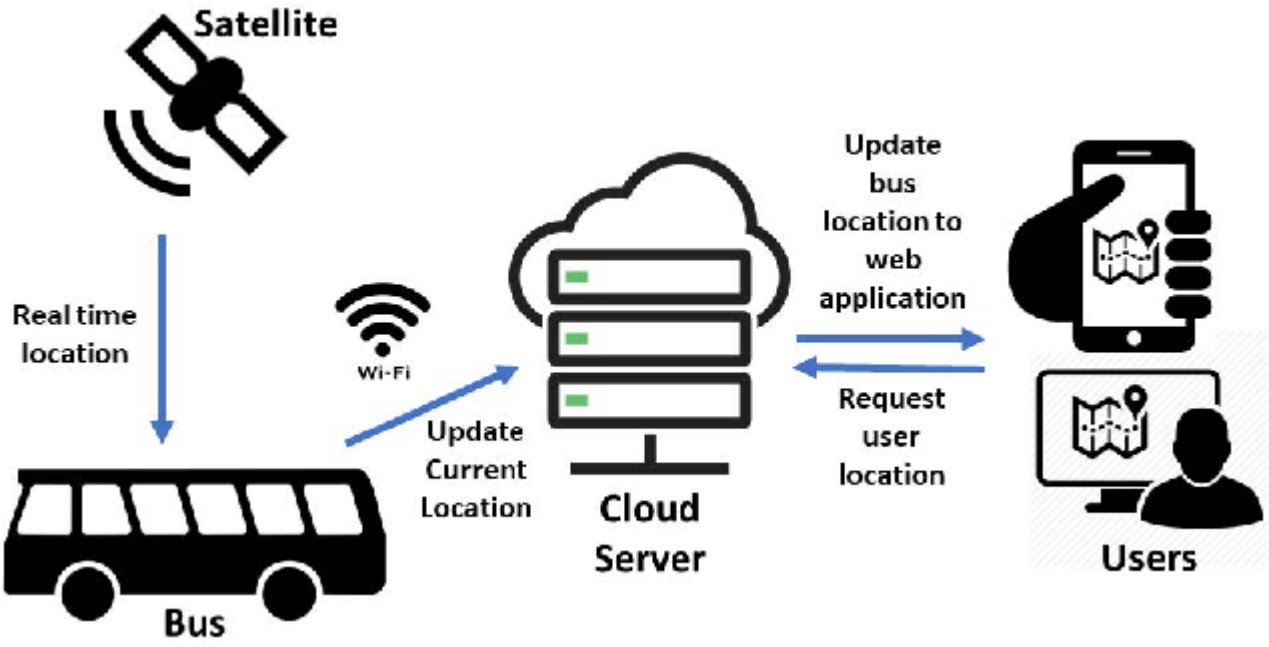
Implement real-time data feeds and GPS tracking systems to monitor buses and provide passengers with accurate arrival times and service updates.

14. Testing and Simulation:

Simulate route options and schedules to assess their efficiency and reliability, making adjustments as needed.

15. Feedback Mechanisms:

Establish mechanisms for passengers to provide feedback on routes, stops, and overall service quality.



CONCLUSION:

This program is a basic outline and a complete solution for the public transport optimization .You will need to implement the necessary functionality in the JavaScript file (script.js) to retrieve traffic data, optimize routes, and display them on the map.