### PHASE 4 : IoT - SMART PUBLIC RESTROOM

### To create a platform displaying real-time restroom availability and cleanliness data using web development technologies,

### 1. Frontend Development:

### HTML:

### Create the structure of your web page. For example:

### #Program

### <!DOCTYPE html>

### <html lang="en">

### <head>

### <meta charset="UTF-8">

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

### <title>Restroom Availability</title>

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

### </head>

### <body>

### <div class="restroom-container" id="restroom-container">

### <!-- Restroom data will be dynamically inserted here -->

### </div>

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

### </body>

### </html>

### CSS (styles.css):

### Style your web page for a better user experience. For example:

### #Program

### body {

### font-family: Arial, sans-serif;

### background-color: #f2f2f2;

### margin: 0;

### padding: 0;

### .restroom-card {

### background-color: white;

### border-radius: 10px;

### box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

### margin: 20px;

### padding: 20px;

### }

### .restroom-card h2 {

### margin-bottom: 10px;

### }

### .restroom-card p {

### margin: 0;

### }

### JavaScript (script.js):

### Use JavaScript to fetch real-time data from the server and update the HTML dynamically. You can use AJAX, Fetch API, or libraries like Axios for making API requests. For example, using Fetch API:

### #Program

### const restroomContainer = document.getElementById("restroom-container");

### function fetchRestroomData() {

### fetch("/api/restrooms") // Assuming your API endpoint is /api/restrooms

### .then(response => response.json())

### .then(data => {

### // Clear previous data

### restroomContainer.innerHTML = "";

### // Iterate through restroom data and create cards

### data.forEach(restroom => {

### const card = document.createElement("div");

### card.classList.add("restroom-card");

### card.innerHTML = `

### <h2>${restroom.name}</h2>

### <p>Availability: ${restroom.availability}</p>

### <p>Cleanliness: ${restroom.cleanliness}</p>

### `;

### restroomContainer.appendChild(card);

### });

### })

### .catch(error => {

### console.error("Error fetching restroom data:", error);

### });

### }

### // Fetch data initially and set interval for periodic updates

### fetchRestroomData();

### setInterval(fetchRestroomData, 5000); // Update every 5 seconds (adjust as needed)

### 2. ****Backend Development:****

### You would need a backend server to handle API requests and provide real-time data. Here's a simple example using Node.js and Express.js:

### #Program

### const express = require("express");

### const app = express();

### // Sample restroom data (replace this with data from your database)

### const restrooms = [

### { id: 1, name: "Restroom 1", availability: "Available", cleanliness: "Clean" },

### { id: 2, name: "Restroom 2", availability: "Occupied", cleanliness: "Slightly Dirty" },

### ];

### app.use(express.json());

### // API endpoint to get restroom data

### app.get("/api/restrooms", (req, res) => {

### res.json(restrooms);

### });

### const PORT = 3000;

### app.listen(PORT, () => {

### console.log(`Server is running on port ${PORT}`);

### });

### In this example, the server provides restroom data when a GET request is made to the /api/restrooms endpoint.

### 3. ****Testing and Deployment:****

### ****Testing:**** Test your web application to ensure that real-time data updates are displayed correctly. Handle potential errors gracefully.

### ****Deployment:**** Deploy your backend server and frontend files on a hosting platform (such as Heroku, Netlify, or Vercel) to make your platform publicly accessible.

### By following these steps, you can create a platform that displays real-time restroom availability and cleanliness data using web development technologies. Make sure to customize the code and styles according to your specific requirements and integrate a suitable backend system for fetching real-time data.

### Mobile App Design:

### User Interface (UI) Design:

### Clean and Intuitive UI: Design a user-friendly interface with easy navigation and intuitive controls.

### Real-time Updates : Implement real-time data updates for restroom availability and cleanliness.

### Maps Integration: Integrate Google Maps or any other mapping service to display restroom locations.

### Feedback System: Include a feedback feature allowing users to rate and provide feedback on restroom cleanliness.

### Functionality:

### Real-time Data: Fetch real-time restroom data from the server using APIs.

### Maps Integration: Display restroom locations on the map with markers for easy navigation.

### Search Functionality: Implement a search feature allowing users to find nearby restrooms.

### Filters: Provide filters based on cleanliness, availability, accessibility, etc.

### User Reviews: Show user reviews and ratings for each restroom.

### Push Notifications: Optionally, send push notifications for restroom availability updates or new nearby restrooms.

### Technical Implementation:

### iOS App (Swift):

### Use Swift programming language for iOS app development.

### Utilize Xcode IDE for development.

### Implement real-time functionality using WebSockets or Push Notifications.

### Integrate Google Maps SDK for iOS for mapping functionality.

### Android App (Java/Kotlin):

### Use Java or Kotlin programming language for Android app development.

### Utilize Android Studio IDE for development.

### Implement real-time functionality using WebSockets or Push Notifications.

### Integrate Google Maps SDK for Android for mapping functionality.