Name :R Thirulogachander

Reg no:713921106056

NM id :au713921106056

***TITLE: IoT Based Smart Public Restroom***

***Create a platform that displays real-time restroom availability and cleanliness data:***

*Creating a platform that displays real-time restroom availability and cleanliness data involves a comprehensive tech stack, including databases, servers, real-time data processing, and a user interface. I’ll provide a simplified overview and example using Python and Flask for the server-side and HTML/CSS for the front-end. For a production-ready solution, you would typically use a more complex tech stack.*

***1.Server-Side (Python with Flask):***

***From flask import Flask, render\_template, request, jsonify***

***App = Flask(\_\_name\_\_)***

***# Simulated data – Replace with real data sources***

***Restroom\_data = {***

***“restroom\_1”: {***

***“availability”: True,***

***“cleanliness”: 4.5,***

***},***

***“restroom\_2”: {***

***“availability”: False,***

***“cleanliness”: 2.3,***

***},***

***}***

***@app.route(‘/’)***

***Def index():***

***Return render\_template(‘index.html’)***

***@app.route(‘/api/restrooms’, methods=[‘GET’])***

***Def get\_restroom\_data():***

***Return jsonify(restroom\_data)***

***If \_\_name\_\_ == ‘\_\_main\_\_’:***

***App.run(debug=True)***

***2.Front-End (HTML/CSS):***

*Create a simple HTML and CSS page to display the restroom data.*

***<!DOCTYPE html>***

***<html>***

***<head>***

***<title>Restroom Status</title>***

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

***</head>***

***<body>***

***<h1>Real-Time Restroom Data</h1>***

***<div id=”restroom-data”></div>***

***<script>***

***Function updateRestroomData() {***

***Fetch(‘/api/restrooms’)***

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

***.then(data => {***

***Const restroomData = document.getElementById(‘restroom-data’);***

***restroomData.innerHTML = ‘’;***

***for (const [restroom, status] of Object.entries(data)) {***

***const div = document.createElement(‘div’);***

***div.innerHTML = `<strong>${restroom}:</strong> Availability: ${status.availability ? ‘Available’ : ‘Occupied’}, Cleanliness: ${status.cleanliness}`;***

***restroomData.appendChild(div);***

***}***

***});***

***}***

***setInterval(updateRestroomData, 5000); // Update every 5 seconds***

***updateRestroomData(); // Initial update***

***</script>***

***</body>***

***</html>***

*In this simplified example:*

* *We use Flask to create a basic server with endpoints for serving the HTML page and providing real-time restroom data.*
* *The HTML page displays restroom data retrieved from the server using JavaScript’s fetch.*
* *The data is updated every 5 seconds using setInterval.*

***3.Data Sources:***

*In a real-world scenario, you would replace the simulated data in the Flask application with actual data sources, such as IoT sensors, databases, or APIs that provide real-time restroom availability and cleanliness information.*

*This example serves as a starting point, but a production-ready platform would require a more robust architecture, security, scalability, and a user-friendly interface. Additionally, you might consider using real-time technologies like WebSockets for instant updates.*

***Design mobile apps for iOS and Android platforms that provide users with access to realtime restroom information :***

*Designing and developing a mobile app for iOS and Android that provides users with real-time restroom information involves multiple steps, including UI/UX design and programming. Below, I’ll provide a simplified example of the programming aspect using Python and Kivy, a Python library for building multi-touch applications.*

*Please note that for a production-ready app, you would likely use native development (Swift for iOS and Java/Kotlin for Android) or cross-platform frameworks like Flutter or React Native. This example serves as a basic illustration.*

***1.Setup:***

*First, ensure you have Python and Kivy installed on your system. You can install Kivy using pip:*

***Pip install kivy***

***2.Python Code:***

*Here’s a basic structure of the app with Python and Kivy:*

***from kivy.app import App***

***from kivy.uix.boxlayout import BoxLayout***

***from kivy.uix.label import Label***

***from kivy.uix.button import Button***

***class RestroomApp(App):***

***def build(self):***

***layout = BoxLayout(orientation='vertical')***

***self.status\_label = Label(text="Restroom Status: Vacant")***

***self.status\_button = Button(text="Check Status")***

***self.status\_button.bind(on\_press=self.check\_status)***

***layout.add\_widget(self.status\_label)***

***layout.add\_widget(self.status\_button)***

***return layout***

***def check\_status(self, instance):***

***# In a real app, you would fetch real-time data from a server or IoT devices.***

***# Simulate data for demonstration purposes.***

***if self.status\_label.text == "Restroom Status: Vacant":***

***self.status\_label.text = "Restroom Status: Occupied"***

***else:***

***self.status\_label.text = "Restroom Status: Vacant"***

***If \_\_name\_\_ == '\_\_main\_\_':***

***RestroomApp().run()***

*In this example, we’ve created a simple app that displays the restroom status as “Vacant” and allows the user to press a button to change the status to “Occupied” and vice versa. In a real app, you would fetch data from a server or IoT sensors.*

***3.Data Integration:***

*In a production app, you would connect to a server or API to get real-time restroom information. You might use libraries like requests for HTTP requests in Python.*

***4.UI/UX Design:***

*A real app would have a well-designed user interface with screens for different functionalities, such as viewing a map of restroom locations, searching for restrooms, and checking cleanliness and occupancy status.*

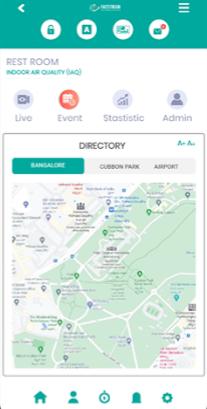
***5.iOS and Android Development:***

*To develop a native app for iOS and Android, you would need to use the respective development environments and programming languages. Swift for iOS and Java/Kotlin for Android are commonly used.*

***6.Testing and Deployment:***

*Thoroughly test your app on both iOS and Android devices. You’ll need developer accounts for both platforms to publish your app to the App Store and Google Play Store.*

*Please note that building a complete and production-ready app involves much more than this basic example, including handling user authentication, database integration, and dealing with real-time data from IoT devices or a server*

********