**SMART PUBLIC RESTROOM**

**IoT – PHASE – 4**

**DEVELOPMENT – 2**

***OBJECTIVES***

* *Creating a website to have real – time monitoring of the restroom.*
* *Publishing an application to receive alerts on the restroom usability and cleanliness.*
* *A cloud service to store and update sensor values in real time.*

***Creating the Website:***

**Website Overview:**

The website is designed for a smart public restroom with IoT sensors. It provides information about three key aspects: "Occupancy Detection," "Hygiene of the Restroom," and "Water Quality and Availability." Each section of the website includes a "Learn More" button that, when clicked, reveals detailed information about what the sensors are reading in real-time.

**HTML (index.html):**

1. ***Header:*** The header section includes the website's title, tagline, and a logo. It also contains a navigation menu to easily access the three key sections.

2. ***Main Content:*** The main content section includes three sections, each dedicated to one of the key aspects: "Occupancy Detection," "Hygiene of the Restroom," and "Water Quality and Availability." Each section has a title and a "Learn More" button that reveals detailed sensor data upon clicking.

3. ***Footer:*** The footer contains "Contact Us" information and social media links with icons.

**CSS (styles.css):**

1. ***Basic Styles:*** The CSS code defines basic styles for the body, header, navigation, main content, and footer. It sets the fonts, colors, and overall layout.

2. ***Hover*** ***Effects:*** It provides hover effects for navigation links and sections, making the website more interactive and visually appealing.

3. ***Footer Styles:*** The CSS styles for the footer, including the "Contact Us" and social media links.

**JavaScript (script.js):**

1. ***Firebase Configuration:*** The JavaScript code includes a configuration for Firebase, a cloud database used to store sensor data.

2. ***Sensor Data Push:*** It defines a function to push sensor data to Firebase and includes an example of how to send sample sensor data to the database.

3. ***Section Interaction:*** Event listeners are added for the "Learn More" buttons in each section. When clicked, they toggle the visibility of detailed sensor data and fetch real-time data from Firebase (in this example, the sample data is updated).

4. ***Update Sensor Data:*** The script includes a function to update the sensor data for each section and display it in real-time. This is where you would replace the sample data with actual data from IoT sensors.

This website provides a user-friendly interface to access information about the smart public restroom, including occupancy, hygiene, and water quality. It's designed to be visually appealing with hover effects, and it offers the ability to view real-time sensor data when clicking on the "Learn More" buttons. The website is powered by Firebase to store and retrieve sensor data, making it a dynamic and informative platform.

|  |
| --- |
| ***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 Restroom</title>  </head>  <body>      <header style="background-image: url('header-background.jpg');">          <img src="logo.png" alt="Smart Public Restroom Logo" id="logo">          <h1 class="hero-title">Smart Public Restroom</h1>          <p class="tagline">Experience Cleanliness and Convenience</p>      </header>      <nav>          <ul>              <li><a href="#occupancy">Occupancy Detection</a></li>              <li><a href="#hygiene">Hygiene</a></li>              <li><a href="#water-quality">Water Quality and Availability</a></li>          </ul>      </nav>      <main>         <section id="occupancy">      <h2>Occupancy Detection</h2>      <p>Learn about the occupancy status in real-time:</p>      <button class="cta-button" id="occupancy-button">Learn More</button>      <div class="details" id="occupancy-details">          <p>Occupancy: <span id="occupancy-status">Loading...</span></p>          <p>Details: <span id="occupancy-details-text">Loading...</span></p>      </div>  </section>          <section id="hygiene">      <h2>Hygiene of the restroom</h2>      <p>Learn about the hygiene status in real-time:</p>      <button class="cta-button" id="hygiene-button">Learn More</button>      <div class="details" id="hygiene-details">          <p>Hygiene: <span id="hygiene-status">Loading...</span></p>          <p>Details: <span id="hygiene-details-text">Loading...</span></p>      </div>  </section>        <section id="water-quality">      <h2>Water Quality Detection</h2>      <p>Learn about the water-quality status in real-time:</p>      <button class="cta-button" id="water-quality-button">Learn More</button>      <div class="details" id="water-quality-details">          <p>Water-Quality: <span id="water-quality-status">Loading...</span></p>          <p>Details: <span id="water-quality-details-text">Loading...</span></p>      </div>  </section>      </main>      <script src="script.js"></script>      <script src="https://www.gstatic.com/firebasejs/9.6.1/firebase-app.js">  </script>  <script src="https://www.gstatic.com/firebasejs/9.6.1/firebase-database.js">  </script>  <footer>      <div class="contact">          <h3>Contact Us</h3>          <p>Email: smartpublicrestroom@gmail.com</p>          <p>Phone: +91 8224532153</p>      </div>      <div class="social">          <h3>Follow Us</h3>          <a href="#" target="\_blank"><img src="facebook.png" alt="Facebook"></a>          <a href="#" target="\_blank"><img src="twitter.png" alt="Twitter"></a>          <a href="#" target="\_blank"><img src="instagram.png" alt="Instagram"></a>      </div>  </footer>  </body>  </html> |

|  |
| --- |
| ***CSS***  body {      font-family: Raleway;      background-color: #0B0B45;      margin: 0;  }  header {      background: linear-gradient(to bottom, #4CAF50, #45a049); /\* Gradient background with bright green colors \*/      color: #fff;      text-align: center;      padding: 50px 0;      position: relative;  }  #logo {      width: 100px;      height: 100px;      margin: 0 auto;      display: block;  }  .hero-title {      font-size: 36px;  }  .tagline {      font-size: 18px;      margin-top: 20px;  }  nav ul {      background-color: #333;      text-align: center;      list-style: none;      padding: 10px 0;  }  nav li {      display: inline;      margin: 0 20px;  }  nav a {      text-decoration: none;      color: #fff;      transition: color 0.2s; /\* Smooth color transition on hover \*/  }  nav a:hover {      color: #ff6b6b; /\* Bright red color on hover \*/  }  main {      padding: 30px;  }  section {      margin: 30px 0;      padding: 20px;      border: 1px solid #ddd;      background-color: #fff;      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);      transition: transform 0.2s, background-color 0.2s; /\* Add transition for transform and background-color \*/  }  section:hover {      transform: scale(1.05); /\* Enlarge the section on hover \*/      box-shadow: 0 0 15px rgba(0, 0, 0, 0.2); /\* Adjust the box shadow on hover \*/      background-color: #ffcc80; /\* Bright orange color on hover \*/      transition: transform 0.2s, background-color 0.2s; /\* Apply transition on hover \*/  }  .cta-button {      display: inline-block;      padding: 10px 20px;      background-color: #ff6b6b; /\* Bright red color for CTA buttons \*/      color: #fff;      text-decoration: none;      border: none;      border-radius: 5px;      margin-top: 10px;      cursor: pointer;      transition: background-color 0.2s; /\* Smooth color transition on hover \*/  }  .cta-button:hover {      background-color: #ffcc80; /\* Bright orange color on CTA buttons when hovered \*/  }  /\* ... (previous styles) ... \*/  footer {      background-color: #333;      color: #fff;      padding: 20px 0;      text-align: center;      display: flex;      justify-content: space-around;  }  .contact, .social {      flex: 1;  }  .contact h3, .social h3 {      font-size: 18px;  }  .contact p, .social a {      font-size: 14px;      margin: 5px 0;  }  .social img {      width: 30px;      height: 30px;      margin: 5px;  } |

|  |
| --- |
| ***JAVASCRIPT***  // Define sample sensor data (you should replace this with actual data)  const sensorData = {      occupancy: {          status: "Occupied",          details: "2 out of 5 stalls are currently in use."      },      hygiene: {          status: "Clean",          details: "Restroom was cleaned 1 hour ago."      },      waterQuality: {          status: "Safe",          details: "Water quality meets safety standards."      }  };  // Function to update sensor data in the sections  function updateSensorData(section) {      const statusElement = document.getElementById(`${section}-status`);      const detailsElement = document.getElementById(`${section}-details-text`);      if (sensorData[section]) {          statusElement.textContent = sensorData[section].status;          detailsElement.textContent = sensorData[section].details;      } else {          statusElement.textContent = "N/A";          detailsElement.textContent = "Data not available.";      }  }  // Add event listeners to "Learn More" buttons  document.getElementById("occupancy-button").addEventListener("click", function () {      const detailsElement = document.getElementById("occupancy-details");      detailsElement.style.display = detailsElement.style.display === "block" ? "none" : "block";      updateSensorData("occupancy");  });  // Repeat the above event listeners for "Hygiene of the Restroom" and "Water Quality and Availability"  // Initial update of sensor data  updateSensorData("occupancy");  document.getElementById("hygiene-button").addEventListener("click", function () {      const detailsElement = document.getElementById("hygiene-details");      detailsElement.style.display = detailsElement.style.display === "block" ? "none" : "block";      updateSensorData("hygiene");  });  // Repeat the above event listeners for "Hygiene of the Restroom" and "Water Quality and Availability"  // Initial update of sensor data  updateSensorData("hygiene");  document.getElementById("waterQuality-button").addEventListener("click", function () {      const detailsElement = document.getElementById("waterQuality-details");      detailsElement.style.display = detailsElement.style.display === "block" ? "none" : "block";      updateSensorData("waterQuality");  });  // Initial update of sensor data  updateSensorData("waterQuality");  // Firebase configuration (replace with your Firebase project configuration)  const firebaseConfig = {      apiKey: "AIzaSyDeeOFMYYxzxvU-0HgbBE\_LWpCq9fZtaxc",      authDomain: "smart-public-restroom-e4268.firebaseapp.com",      databaseURL: "https://smart-public-restroom-e4268-default-rtdb.firebaseio.com",      projectId: "smart-public-restroom-e4268",      storageBucket: "smart-public-restroom-e4268.appspot.com",      messagingSenderId: "454465283663",      appId: "1:454465283663:web:ad1398258ea61b39cc2c3a",      measurementId: "G-NKD07P5F32"    };  // Initialize Firebase  firebase.initializeApp(firebaseConfig);  // Function to push sensor data to Firebase  function pushSensorData(data) {      var database = firebase.database();      var sensorDataRef = database.ref('sensor\_data');      sensorDataRef.push(data);  }  // Example of pushing data to Firebase when a button is clicked  document.getElementById("sensorDataButton").addEventListener("click", function() {      var sensorData = {          occupancy: true,          hygiene: "clean",          waterQuality: "safe",          waterAvailability: "available"      };      pushSensorData(sensorData);  }); |

***Creating an Application:***

1***. Design the User Interface:***

- In MIT App Inventor, use the designer to create your app's user interface.

- Create three buttons for "Occupancy," "Hygiene," and "Water Quality." Set their background color to red initially.

- Add labels to display the status of each section.

2. ***Connect to Firebase:***

- Set up Firebase as explained earlier, obtaining the Firebase configuration details and initializing FirebaseDB.

3. ***Fetch and Display Real-Time Data:***

- In the "Blocks" section of MIT App Inventor:

a. Use the `Screen.Initialize` event to initialize your app.

b. Use the `FirebaseDB.ValueChanged` event to listen for changes in the Firebase database. When data changes, fetch the new data and update the UI elements accordingly.

c. In response to data changes, set the background color of the buttons and update label text based on the sensor data:

- If occupancy is vacant, change the "Occupancy" button's background color to green.

- If hygiene is clean, change the "Hygiene" button's background color to green.

- If water quality is good, change the "Water Quality" button's background color to green.

- Otherwise, keep the buttons' background color red.

4. ***Test Your App:***

- Use the MIT AI Companion app or generate an APK file to install the app on an Android device.

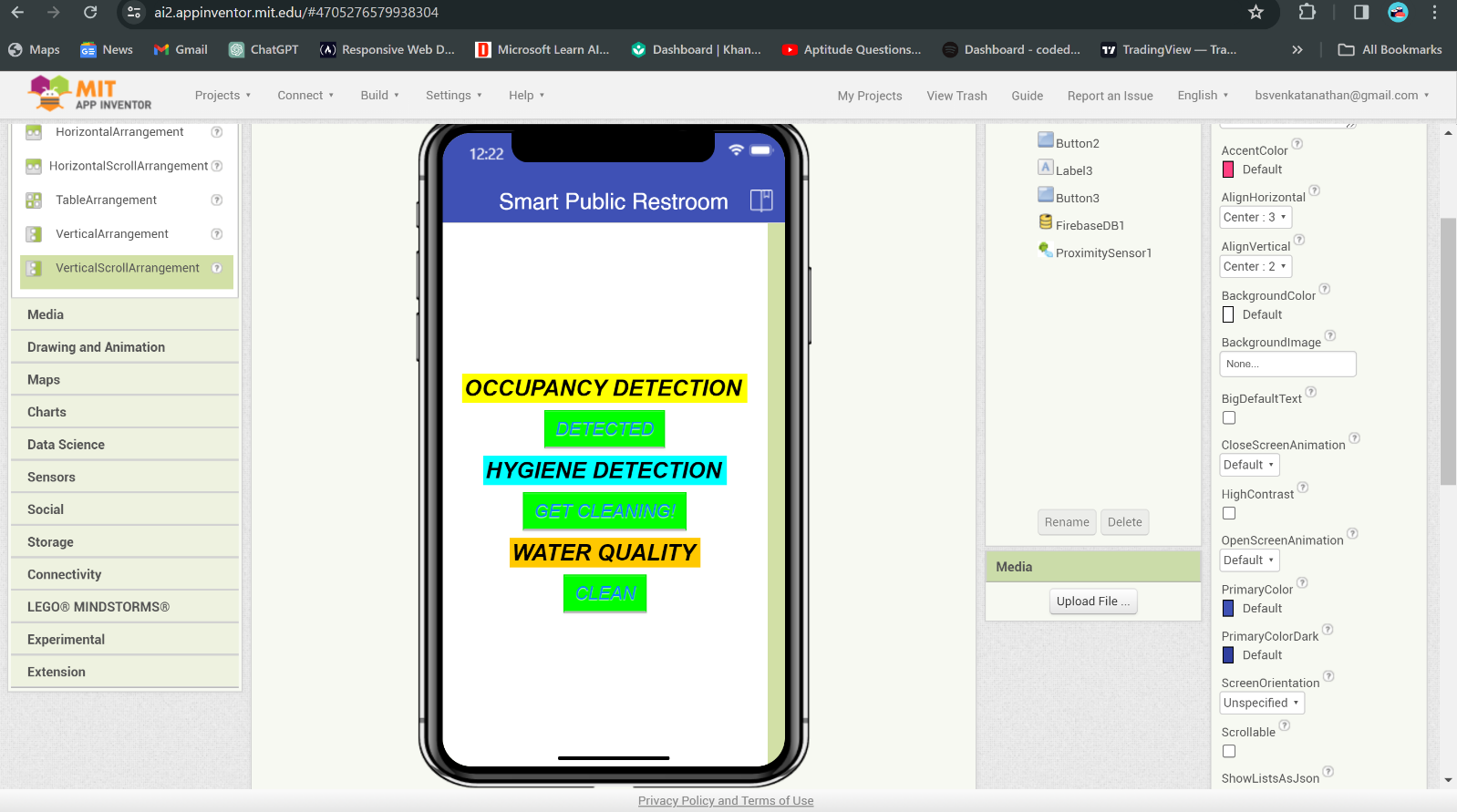
- Verify that the button colors and labels update correctly based on real-time data changes.

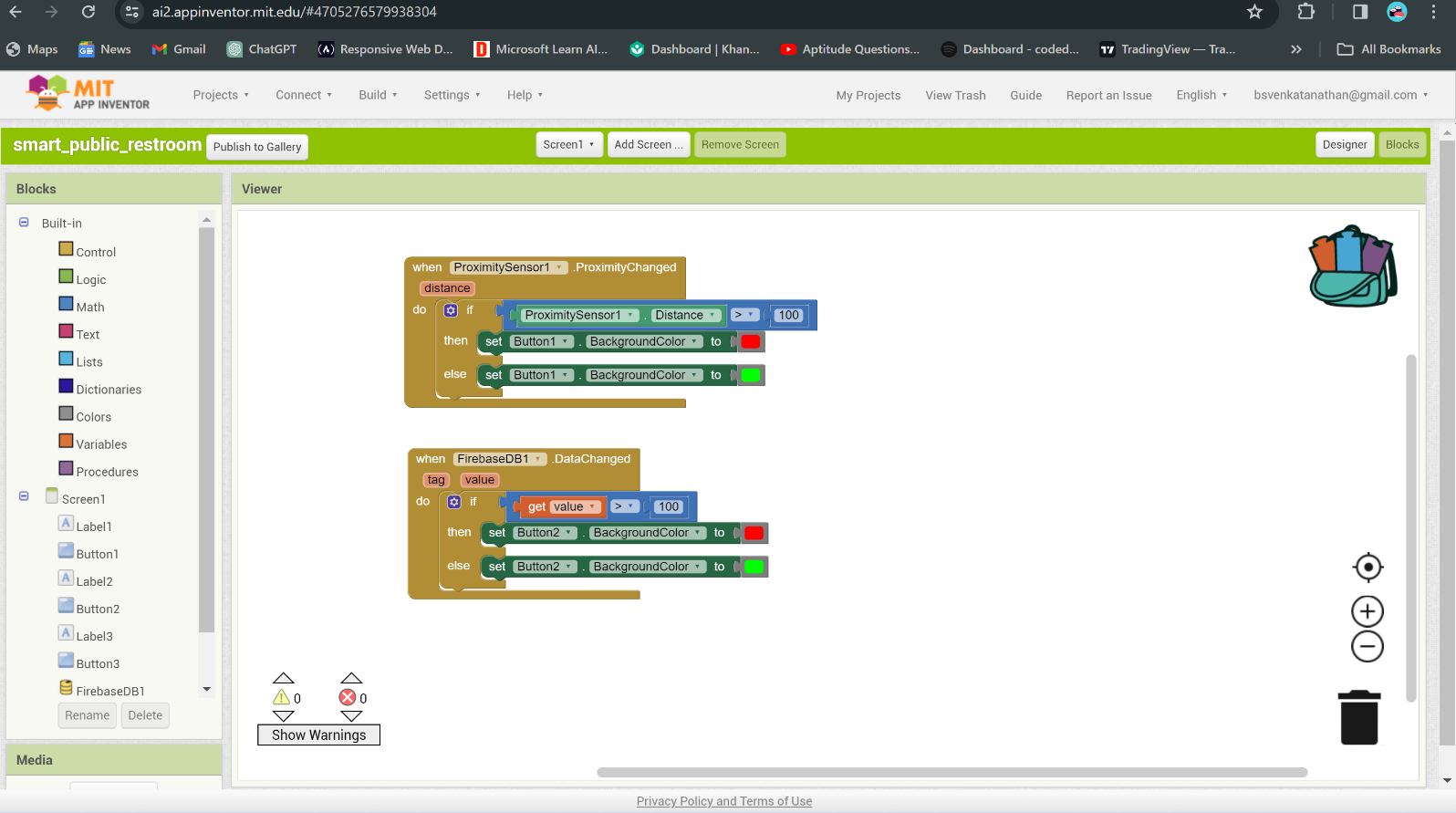
5. ***Additional Features (Optional):***

- Implement further functionality, such as setting up notifications to alert users or staff when certain conditions are met (e.g., low occupancy or poor water quality).

6. ***Deployment:***

- Once your app is tested and ready, you can deploy it to the Google Play Store or distribute it to users.

****

****

***Setting up a Cloud service:***

***Set Up Firebase for MIT App Inventor:***

1. ***Create a Firebase Project:***

- Go to the [Firebase Console](https://console.firebase.google.com/) and create a new Firebase project, providing it with a relevant name related to your Smart Public Restroom project.

2. ***Enable Firebase Realtime Database:***

- In your Firebase project, navigate to the "Realtime Database" section.

- Click on "Create Database" and select "Start in test mode" for development purposes. You can adjust security rules later for production.

3. ***Obtain Firebase Configuration:***

- In your Firebase project, access the project settings (click on the gear icon).

- Under "Your apps," select the appropriate platform (Web/JavaScript) to get your Firebase configuration, which includes details like the API key, database URL, and project ID.

4. ***Configure Firebase in MIT App Inventor:***

- In your existing MIT App Inventor project:

a. ***Add Firebase Components:***

- In the "Designer" section, add the following Firebase components from the "Palette":

- `FirebaseDB` (for the Realtime Database)

b. ***Initialize FirebaseDB:***

- In the "Blocks" section, create an `initialize` block for `FirebaseDB`.

- Enter your Firebase Web API key and database URL using the configuration obtained from your Firebase project settings.