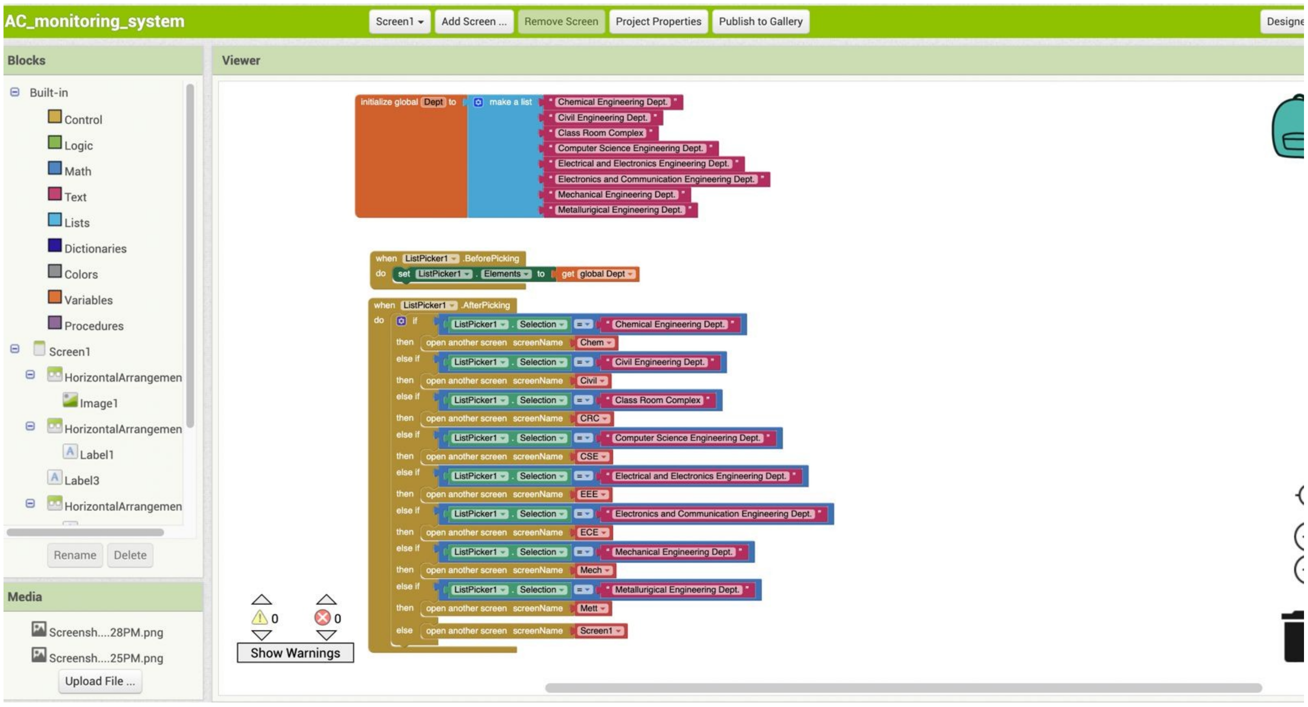
2. APP INTERFACE USING MIT APP INVENTOR:

The app interface was made using the bit app inventor. The home page of the application was developed using MIT App Inventor to provide an intuitive interface for users to navigate to different departmental screens based on their selection. The block coding employed is designed to manage the selection process and ensure that users are redirected to the appropriate department screen efficiently.



```
initialize global Dept to  make a list  "Chemical Engineering Dept."
                                        "Civil Engineering Dept."
                                        "Class Room Complex"
                                        "Computer Science Engineering Dept."
                                        "Electrical and Electronics Engineering Dept."
                                        "Electronics and Communication Engineering Dept."
                                        "Mechanical Engineering Dept."
                                        "Metallurigical Engineering Dept."

when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to  get global Dept

when ListPicker1 .AfterPicking
do  if      ListPicker1 . Selection = "Chemical Engineering Dept."
    then    open another screen screenName Chem
    else if ListPicker1 . Selection = "Civil Engineering Dept."
    then    open another screen screenName Civil
    else if ListPicker1 . Selection = "Class Room Complex"
    then    open another screen screenName CRC
    else if ListPicker1 . Selection = "Computer Science Engineering Dept."
    then    open another screen screenName CSE
    else if ListPicker1 . Selection = "Electrical and Electronics Engineering Dept."
    then    open another screen screenName EEE
    else if ListPicker1 . Selection = "Electronics and Communication Engineering Dept."
    then    open another screen screenName ECE
    else if ListPicker1 . Selection = "Mechanical Engineering Dept."
    then    open another screen screenName Mech
    else if ListPicker1 . Selection = "Metallurigical Engineering Dept."
    then    open another screen screenName Mett
    else    open another screen screenName Screen1
```

Figure 3.2.1

Select:

It is a listPicker and allows to pick the department to be monitored. The options of the ListPicker are initialized through the 'initialize global Dept to make a list' i.e A global variable named Dept is initialized to store a list of department names. This global list serves as the source for populating the ListPicker component, allowing users to select a department from a predefined set of options.

when ListPicker1.BeforePicking do:

i.e Prior to user interaction with the ListPicker, this block ensures that the elements displayed within the ListPicker1 dropdown are set to the values contained in the global Dept list. This guarantees that the user is presented with the correct and complete list of department options each time they initiate a selection.

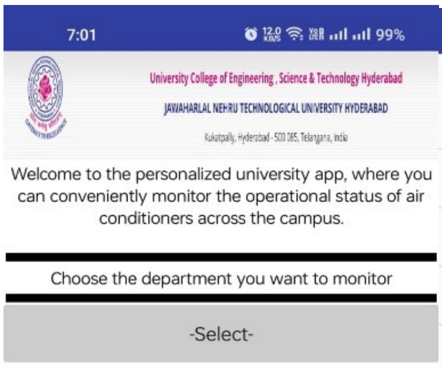
when ListPicker1.AfterPicking do:

Once a user makes a selection from the ListPicker, the application processes the selection and navigates to the corresponding departmental screen. This is achieved through a series of conditional blocks that compare the user's selection against the list of departments.

- If "Chemical Engineering Dept." is selected, the application opens the screen named Chem.
- If "Civil Engineering Dept." is selected, the application opens the screen named Civil.
- If "Class Room Complex" is selected, the application opens the screen named CRC.

- If "Computer Science Engineering Dept." is selected, the application opens the screen named CSE.
- If "Electrical and Electronics Engineering Dept." is selected, the application opens the screen named EEEE.
- If "Electronics and Communication Engineering Dept." is selected, the application opens the screen named ECE.
- If "Mechanical Engineering Dept." is selected, the application opens the screen named Mech.
- If "Metallurgical Engineering Dept." is selected, the application opens the screen named Met.
- **Fallback Scenario:** In the event that none of the specified departments are selected, the application defaults to reopening the home screen, labeled as Screen1.

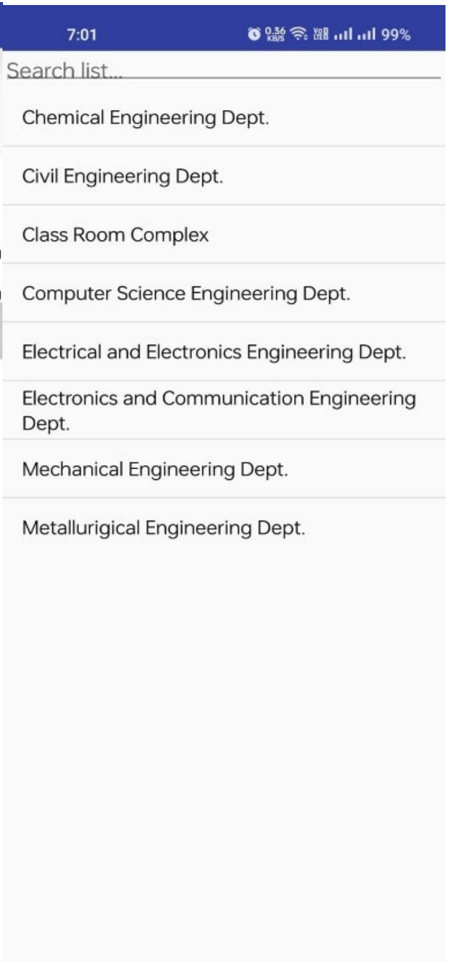
This logical structure ensures that users are seamlessly directed to the appropriate section of the application, based on their initial choice from the list.


7:01
99%
University College of Engineering, Science & Technology Hyderabad
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
Kukatpally, Hyderabad - 500 085, Telangana, India

Welcome to the personalized university app, where you can conveniently monitor the operational status of air conditioners across the campus.

Choose the department you want to monitor

-Select-

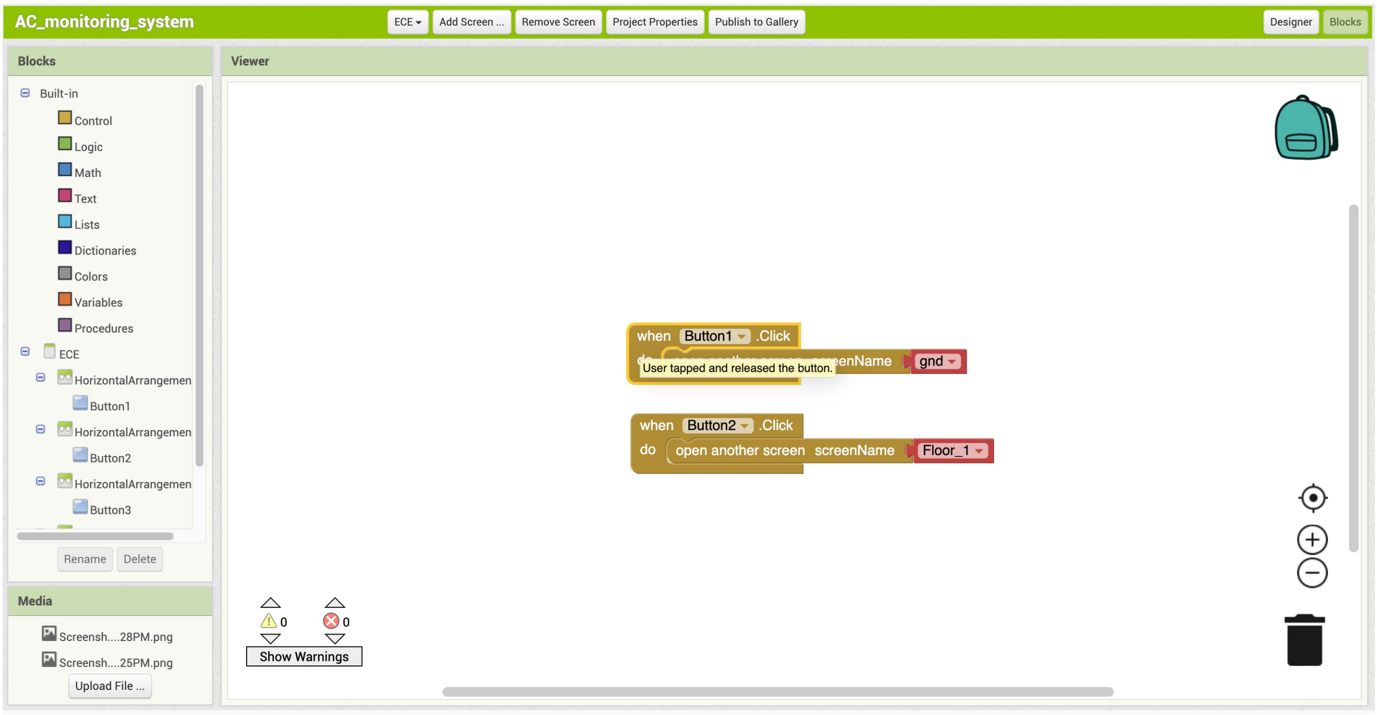


Chemical Engineering Dept.
Civil Engineering Dept.
Class Room Complex
Computer Science Engineering Dept.
Electrical and Electronics Engineering Dept.
Electronics and Communication Engineering Dept.
Mechanical Engineering Dept.
Metallurigical Engineering Dept.

Figure 3.2.2 & Figure 3.2.3

3. ECE DEPARTMENT SCREEN:

After navigating to the ECE department the various floors are listed out and the user can choose which floor is to be monitored. Upon selecting individual floors, the user is redirected to the particular floor screen and details about every particular room/cabin having an air conditioner and its working status are presented. For now, only the' HOD Cabin' in the Ground floor and the room F11 are presented.

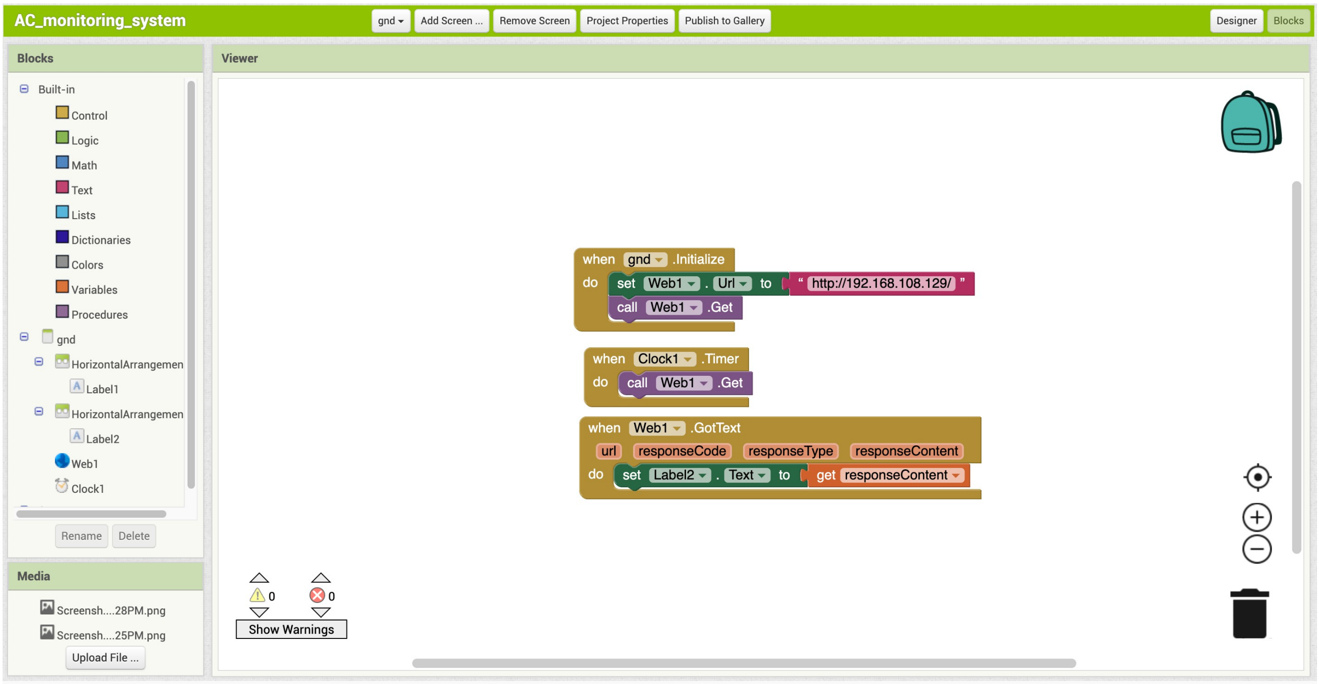


```
when Button1 .Click
do open another screen screenName "gnd"
User tapped and released the button.

when Button2 .Click
do open another screen screenName "Floor_1"
```

Figure 3.3.1

Button1 Click Event: When Button1 is clicked, the app opens the screen named gnd, which is likely representing the Ground Floor.
Button2 Click Event: When Button2 is clicked, the app opens the screen named Floor_1, representing the first floor.



```
when gnd .Initialize
do  set Web1 . Url to " http://192.168.108.129/ "
    call Web1 .Get

when Clock1 .Timer
do  call Web1 .Get

when Web1 .GotText
  url  responseCode  responseType  responseContent
do  set Label2 . Text to get responseContent
```

Figure 3.3.2

Initialize Event:

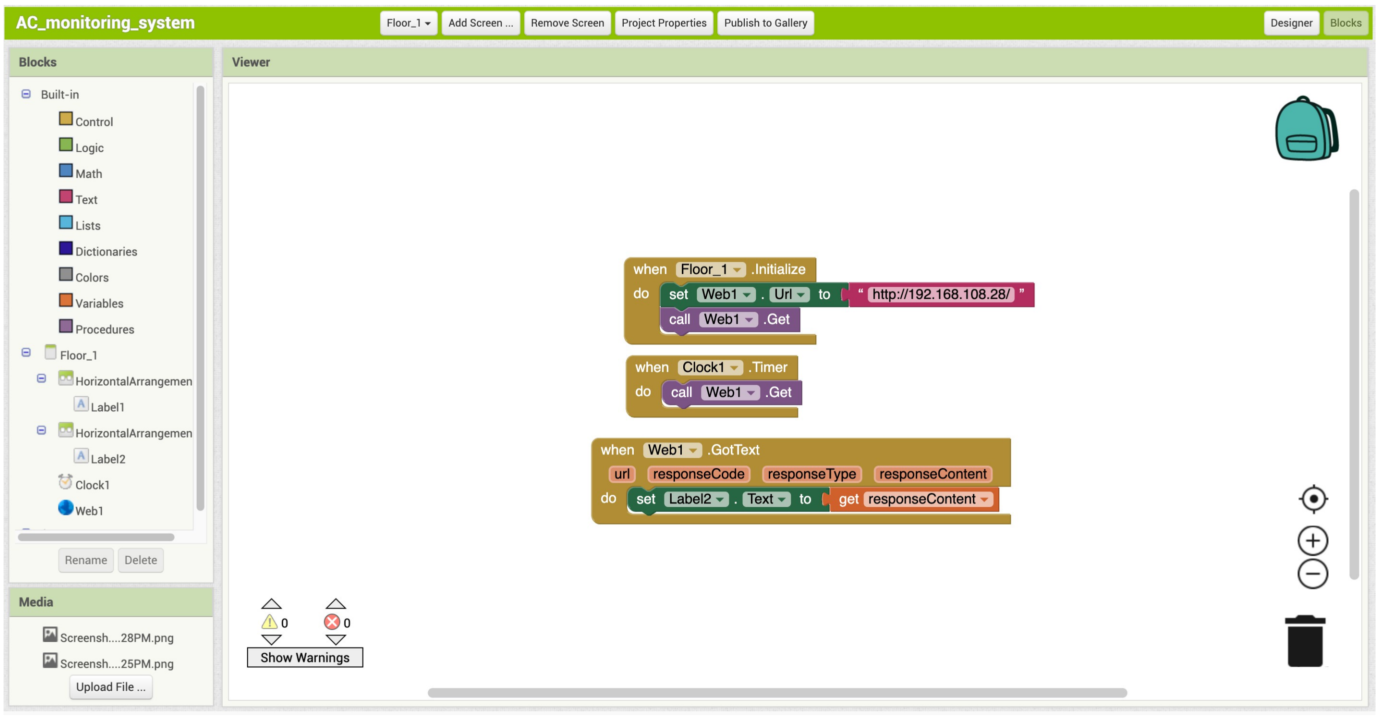
When the ground floor screen initializes, the URL of Web1 is set to `http://192.168.108.28/`, which is the IP address of the Node MCU. This allows the app to connect to the Node MCU for data retrieval. The `Web1.Get` method is called to fetch data from this URL, starting the communication process.

Clock1 Timer Event:

The `Clock1.Timer` event triggers periodically to call `Web1.Get`, ensuring that the app fetches updated information from the HOD cabin URL. This setup allows users to receive real-time updates without needing to manually refresh the app. It maintains a continuous flow of data, keeping the information current.

Web1 GotText Event:

When the app receives data from the HOD cabin, the `Web1.GotText` event is triggered. This event updates the text of Label2 to display the content fetched from the Node MCU. It ensures that users have immediate access to the latest information, enhancing the app's responsiveness.



```
when Floor_1 .Initialize
do  set Web1 . Url to " http://192.168.108.28/ "
    call Web1 .Get

when Clock1 .Timer
do  call Web1 .Get

when Web1 .GotText
  url  responseCode  responseType  responseContent
do  set Label2 . Text to get responseContent
```

Figure 3.3.3

Initialize Event:

When the First floor screen initializes, the URL of Web1 is set to `http://192.168.108.129/`, which is the IP address of the Node MCU. This allows the app to connect to the Node MCU for data retrieval. The `Web1.Get` method is called to fetch data from this URL, starting the communication process.

Clock1 Timer Event:

The `Clock1.Timer` event triggers periodically to call `Web1.Get`, ensuring that the app fetches updated information from the F11 URL. This setup allows users to receive real-time updates without needing to manually refresh the app. It maintains a continuous flow of data, keeping the information current.

Web1 GotText Event:

When the app receives data from the F11, the `Web1.GotText` event is triggered. This event updates the text of Label2 to display the content fetched from the Node MCU. It ensures that users have immediate access to the latest information, enhancing the app's responsiveness.

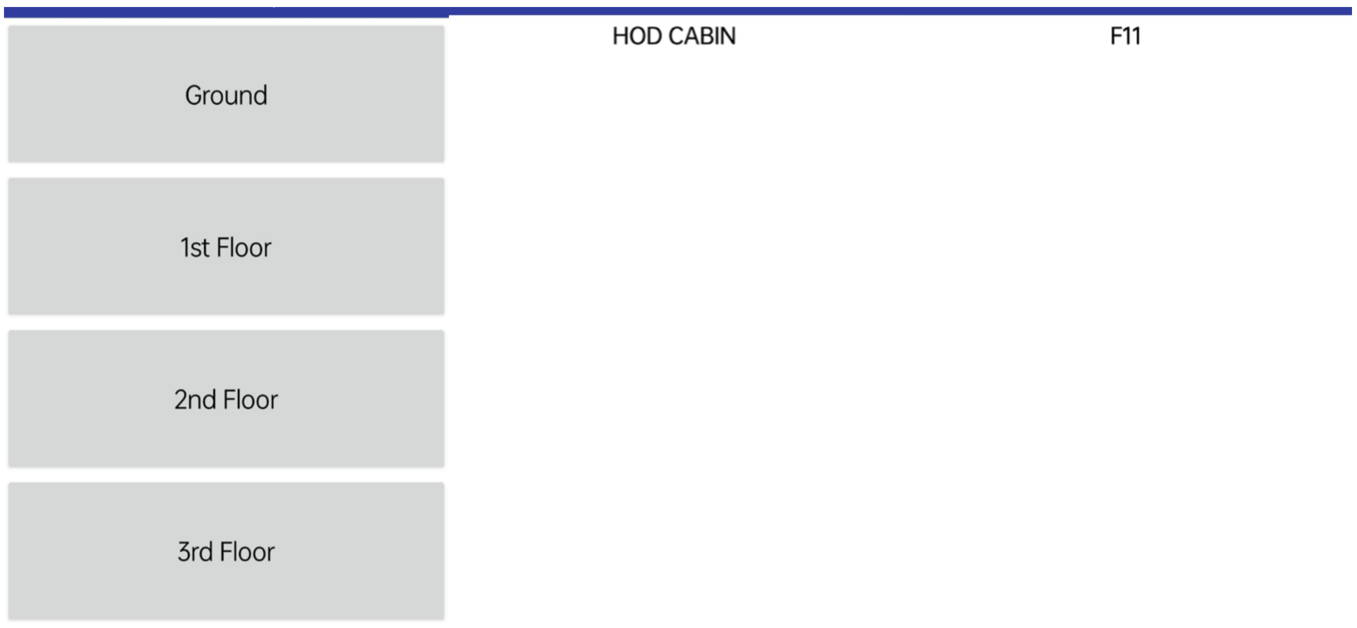


Figure 3.3.4

4. SOFTWARE SERIALAND WCS LIBRARY:

In the Arduino step, Software Serial communication is used to facilitate the transfer of data from the Arduino to the Node MCU. The Arduino, equipped with the WCS1700 current sensor, collects data regarding the current status of the air conditioning system. This data is then transmitted via Software Serial, ensuring effective serial communication between the two devices.

Once the Node MCU receives the data from the Arduino, it connects to a local Wi-Fi network. The Node MCU hosts a web server, where the received data is dynamically transmitted to an HTML website. The website is accessible via the Node MCU's local IP address—which, in this case, corresponds to the URLs (http://192.168.108.129/ for the HOD cabin and http://192.168.108.28/ for the F11 room) used within the block coding in the mobile application.

The website presents the real-time data, allowing users to view the status of the air conditioning system remotely. The mobile app built using MIT App Inventor further integrates with this system. Through periodic requests to the respective local IP addresses, the app fetches the latest data from the website and displays it to the user. This ensures seamless monitoring of the AC system both on the web interface and within the mobile application.