Home Sensor Data

Home1_room_sensor:

It generates random room data such as temperature, pressure, window status, AC status, light status, room door status, and person detection status.

Code Overview:

Imports and Setup:

- Import the **MongoClient** class from the **mongodb** package.
- Define the MongoDB connection URI.
- Create a new instance of **MongoClient** using the URI.

Main Function:

- An asynchronous function serving as the entry point.
- Connects to the MongoDB client.
- Specifies the database name as "room_data" and retrieves the corresponding database object.
- Specifies the collection name as "room" and retrieves the corresponding collection object.

Generating Room Data:

- **getRoomData** function generates simulated room data.
- Data includes properties such as temperature, pressure, device statuses, and room conditions.

Inserting Room Data:

- **insertRoomData** function inserts the generated room data into the MongoDB collection.
- Uses the **insertOne** method to add a single document containing room data and a timestamp.

Random Data Generation Functions:

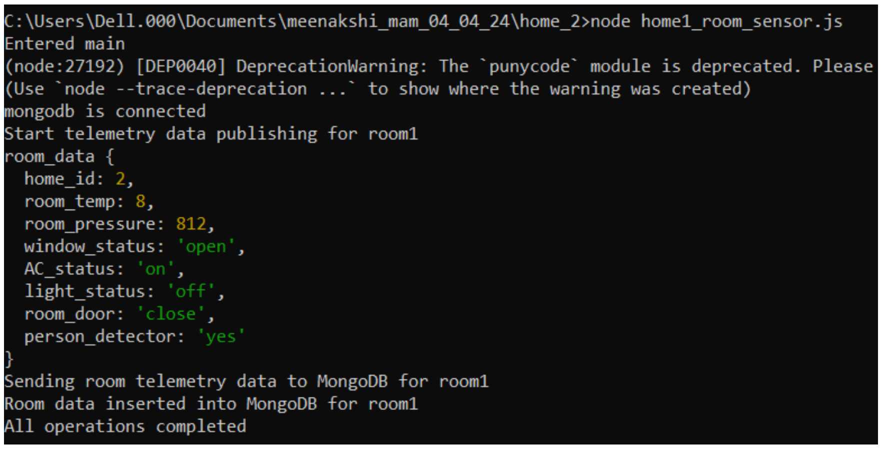
- Helper functions (**generateRandomTemperature**, **generateRandomPressure**, **generateRandomStatus**) generate random values for room properties.

Exporting the Main Function:

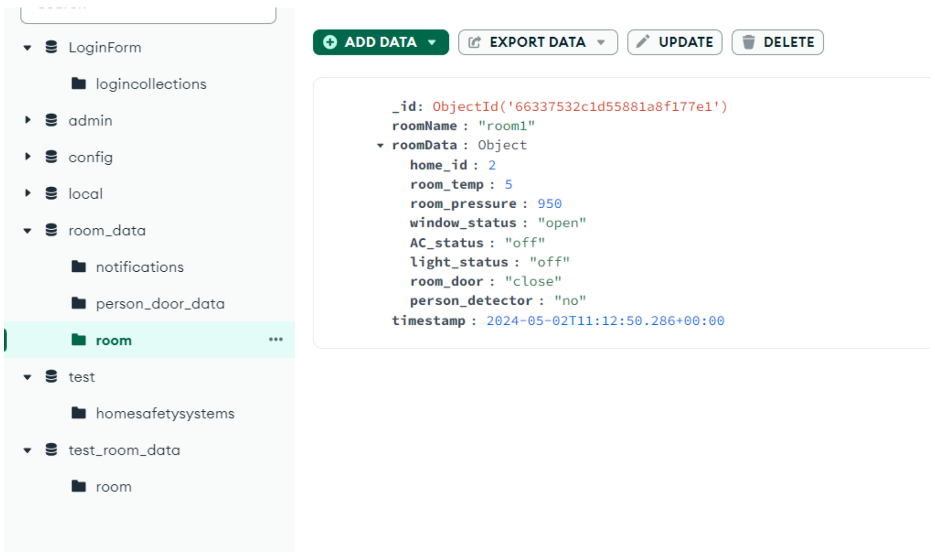
- The **main** function is exported as a module for external use.

Conclusion:

This script facilitates the simulation of telemetry data publishing for a room environment. It establishes a connection to a MongoDB database, generates random room data, and inserts the data into a specified collection. The modular structure allows easy integration into other projects requiring MongoDB telemetry data handling.



```
C:\Users\Dell.000\Documents\meenakshi_mam_04_04_24\home_2>node home1_room_sensor.js
Entered main
(node:27192) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please
(Use `node --trace-deprecation ...` to show where the warning was created)
mongodb is connected
Start telemetry data publishing for room1
room_data {
  home_id: 2,
  room_temp: 8,
  room_pressure: 812,
  window_status: 'open',
  AC_status: 'on',
  light_status: 'off',
  room_door: 'close',
  person_detector: 'yes'
}
Sending room telemetry data to MongoDB for room1
Room data inserted into MongoDB for room1
All operations completed
```



```
_id: ObjectId('66337532c1d55881a8f177e1')
roomName : "room1"
roomData : Object
  home_id : 2
  room_temp : 5
  room_pressure : 950
  window_status : "open"
  AC_status : "off"
  light_status : "off"
  room_door : "close"
  person_detector : "no"
timestamp : 2024-05-02T11:12:50.286+00:00
```

Temp_ac_lambda.js

This file represents a lambda function responsible for controlling the AC based on room temperature. It retrieves room temperature data from the MongoDB database, converts it to Fahrenheit, and determines whether to switch the AC on or off.

Code Overview:

Imports and Setup:

- Import the **MongoClient** class from the **mongodb** package.
- Define the MongoDB connection URI.
- Create a new instance of **MongoClient** using the URI.

Main Function:

- An asynchronous function serving as the entry point.
- Connects to the MongoDB client.
- Specifies the database name as "room_data" and retrieves the corresponding database object.
- Specifies the "room" collection and fetches all records using **find({})** and **toArray()**.

Data Processing Loop:

- Iterates through each record fetched from the "room" collection.
- Extracts relevant data such as **id**, **roomTempCelsius**, **roomPressure**, **personDetector**, **lightStatus**, and **roomdoor**.
- Calculates the room temperature in Fahrenheit using the **celsiusToFahrenheit** function.
- Determines the AC status based on the room temperature and a predefined threshold.
- Inserts processed data into the "person_door_data" collection using the **insertData** function.

Conversion Function:

- **celsiusToFahrenheit** function converts Celsius temperature to Fahrenheit.

Data Insertion Function:

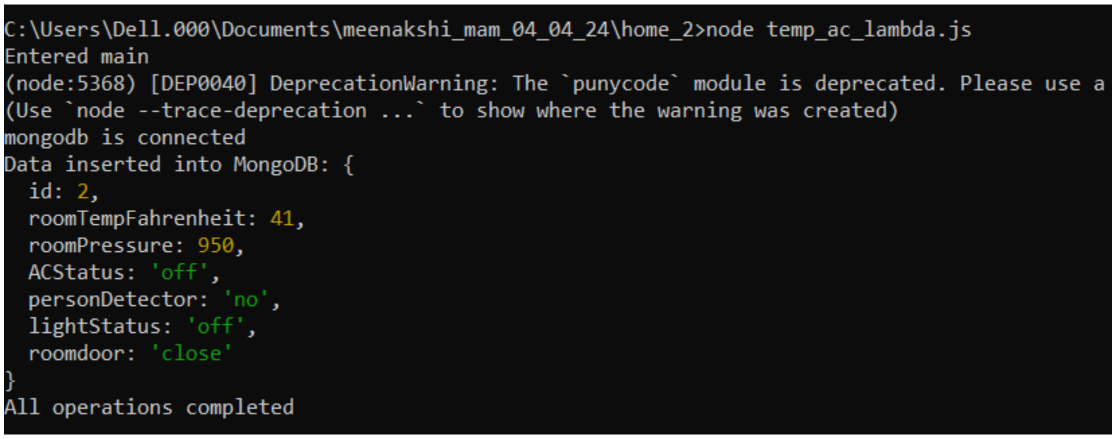
- **insertData** function inserts processed data into the specified collection.
- Uses the **insertOne** method to add a single document containing processed data and a timestamp.

Exporting Functions:

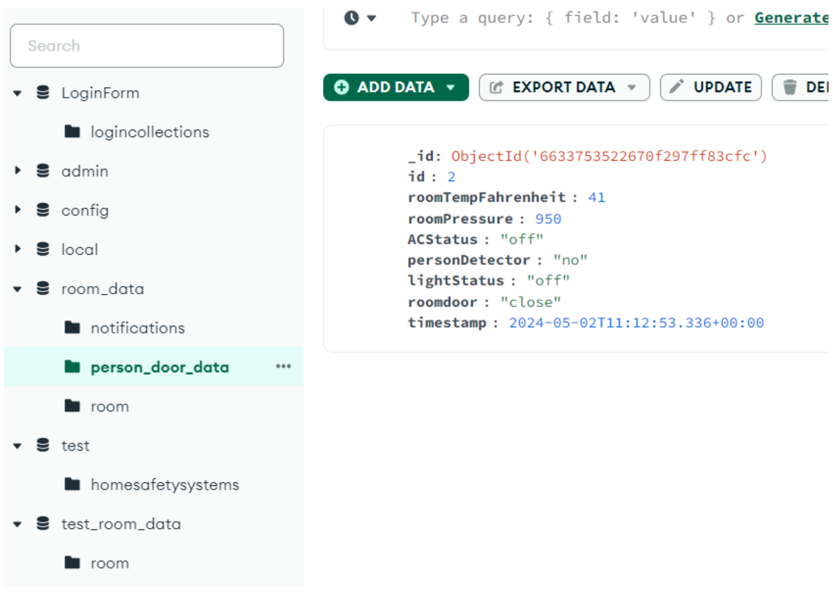
- Exports the **main** function and the **celsiusToFahrenheit** function for external use.

Conclusion:

This script processes telemetry data from a MongoDB collection, calculates AC status based on room temperature, and stores the processed data in a new collection. It demonstrates data manipulation and storage capabilities using MongoDB, providing a foundation for further data analysis and application development.



```
C:\Users\Dell.000\Documents\meenakshi_mam_04_04_24\home_2>node temp_ac_lambda.js
Entered main
(node:5368) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a
(Use `node --trace-deprecation ...` to show where the warning was created)
mongodb is connected
Data inserted into MongoDB: {
  id: 2,
  roomTempFahrenheit: 41,
  roomPressure: 950,
  ACStatus: 'off',
  personDetector: 'no',
  lightStatus: 'off',
  roomdoor: 'close'
}
All operations completed
```



```
_id: ObjectId('6633753522670f297ff83cfc')
id : 2
roomTempFahrenheit : 41
roomPressure : 950
ACStatus : "off"
personDetector : "no"
lightStatus : "off"
roomdoor : "close"
timestamp : 2024-05-02T11:12:53.336+00:00
```

NotifyLambda.js

This file represents a lambda function responsible for generating notifications based on room data. It retrieves room data from the MongoDB database, processes it to determine the appropriate notification message, and inserts the message into the "notifications" collection.

Code Overview:

Imports and Setup:

- Import the **MongoClient** class from the **mongodb** package.
- Define the MongoDB connection URI.
- Create a new instance of **MongoClient** using the URI.

Main Function:

- An asynchronous function serving as the entry point.
- Connects to the MongoDB client.
- Specifies the database name as "room_data" and retrieves the corresponding database object.
- Specifies the "person_door_data" collection and fetches all records using **find({})** and **toArray()**.

Notification Processing Loop:

- Iterates through each record fetched from the "person_door_data" collection.
- Extracts relevant data such as **personDetector**, **ACStatus**, and **doorStatus**.
- Based on the conditions, generates a notification message (**message**).
- Inserts the notification message into the "notifications" collection using the **insertNotification** function.

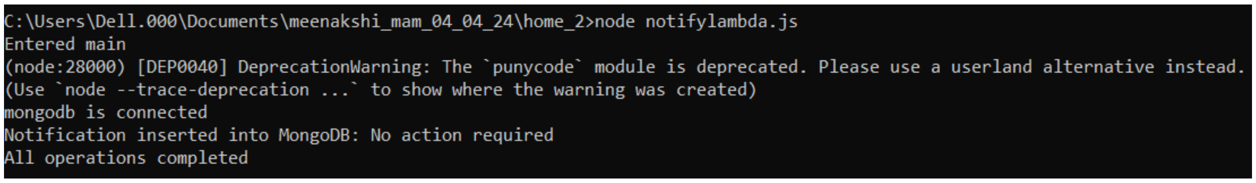
Notification Insertion Function:

- **insertNotification** function inserts the notification message into the specified collection along with a timestamp.
- Uses the **insertOne** method to add a single document containing the message and timestamp.

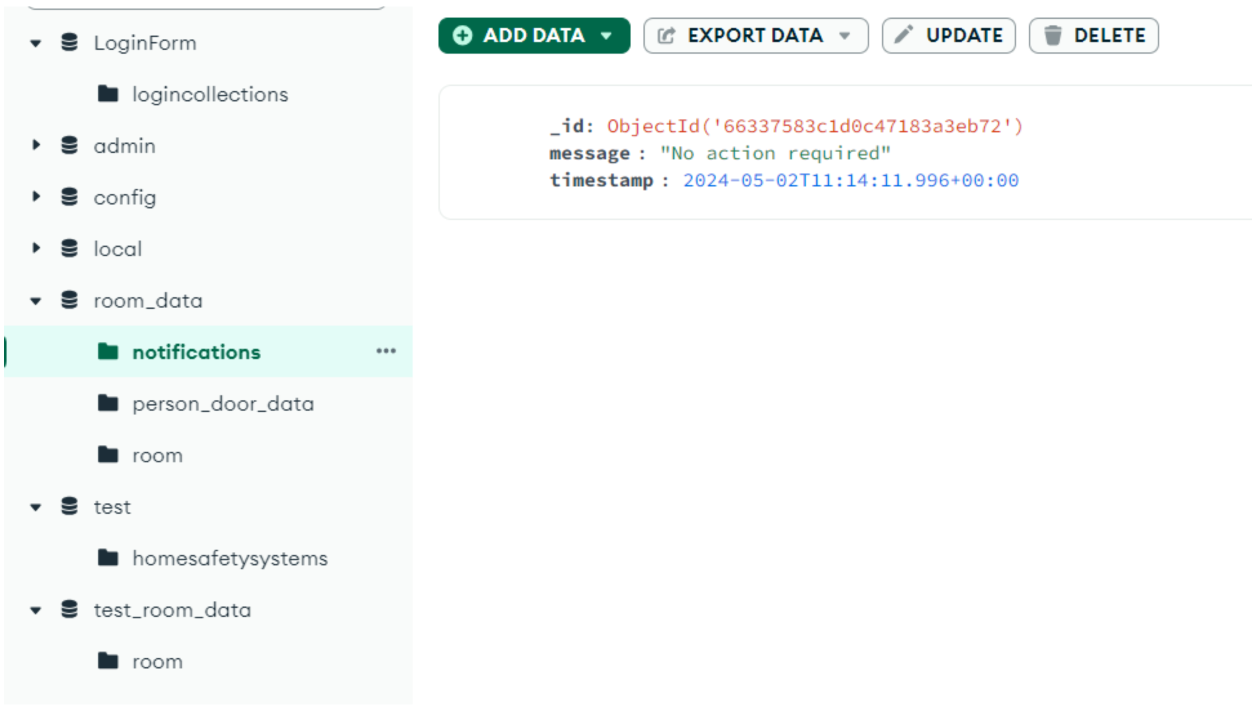
Exporting Function:

- Exports the **main** function for external use.

Conclusion: This script processes data from a MongoDB collection, generates notification messages based on certain conditions, and stores the notifications in another collection. It demonstrates how to implement notification systems using MongoDB, enabling the management of real-time events and actions.



```
C:\Users\Dell.000\Documents\meenakshi_mam_04_04_24\home_2>node notifylambda.js
Entered main
(node:28000) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
mongodb is connected
Notification inserted into MongoDB: No action required
All operations completed
```



```
_id: ObjectId('66337583c1d0c47183a3eb72')
message : "No action required"
timestamp : 2024-05-02T11:14:11.996+00:00
```

Test.js

Integration Tests for Home Automation System

Objective: Perform integration tests for different components of a home automation system, including temperature monitoring, AC control, and notification generation based on room data.

Test Overview:

Setup and Teardown:

- Before running tests, establish a connection to the MongoDB database.
- After all tests are executed, close the MongoDB connection.

Temperature and AC Status Test (temp_ac_lambda):

- **Description:** Checks if the AC status returned by the **tempACMain** function matches the expected status based on the room temperature.

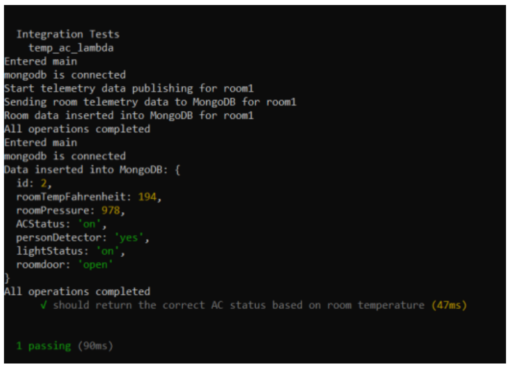
- **Steps:**
    - Retrieve room data from the **home1_room_sensor**.
    - Calculate the expected AC status based on the room temperature converted to Fahrenheit.
    - Call the **tempACMain** function to get the actual AC status.
    - Assert that the actual AC status matches the expected status.

Notification Test (notifylambda):

- **Description:** Verifies if notification messages are correctly inserted into the "notifications" collection based on room data.
- **Steps:**
    - Create a test database and collection named "person_door_data" to simulate room data.
    - Run the **notifyMain** function to generate notifications.
    - Fetch the first notification from the "notifications" collection.
    - Retrieve room data from the "person_door_data" collection to determine the expected notification message.
    - Assert that the actual notification message matches the expected message.
    - Clean up test data by deleting documents from the "person_door_data" and "notifications" collections.

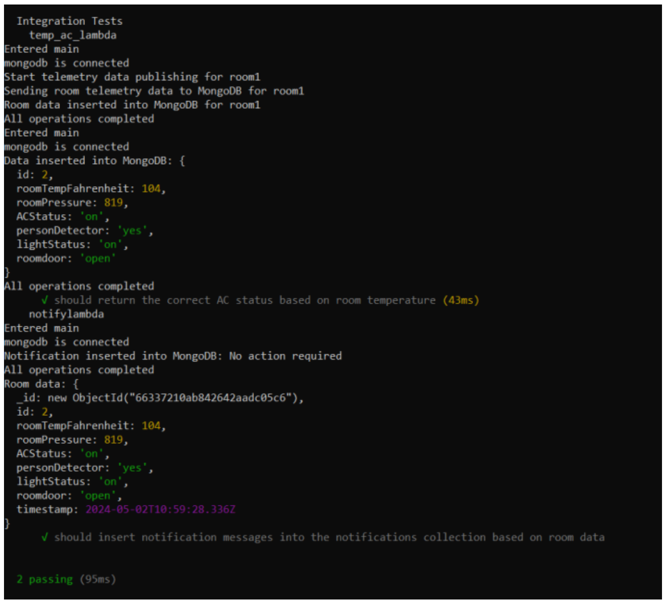
This test suite ensures that the components of the home automation system, such as temperature monitoring and notification generation, function correctly and integrate seamlessly with the MongoDB database.

Console Outputs for 2 test Cases:



```
  Integration Tests
    temp_ac_lambda
Entered main
mongodb is connected
Start telemetry data publishing for room1
Sending room telemetry data to MongoDB for room1
Room data inserted into MongoDB for room1
All operations completed
Entered main
mongodb is connected
Data inserted into MongoDB: {
  id: 2,
  roomTempFahrenheit: 194,
  roomPressure: 978,
  ACStatus: 'on',
  personDetector: 'yes',
  lightStatus: 'on',
  roomdoor: 'open'
}
All operations completed
      √ should return the correct AC status based on room temperature (47ms)


  1 passing (90ms)
```



```
  Integration Tests
    temp_ac_lambda
Entered main
mongodb is connected
Start telemetry data publishing for room1
Sending room telemetry data to MongoDB for room1
Room data inserted into MongoDB for room1
All operations completed
Entered main
mongodb is connected
Data inserted into MongoDB: {
  id: 2,
  roomTempFahrenheit: 104,
  roomPressure: 819,
  ACStatus: 'on',
  personDetector: 'yes',
  lightStatus: 'on',
  roomdoor: 'open'
}
All operations completed
      √ should return the correct AC status based on room temperature (43ms)
    notifylambda
Entered main
mongodb is connected
Notification inserted into MongoDB: No action required
All operations completed
Room data: {
  _id: new ObjectId("66337210ab842642aadc05c6"),
  id: 2,
  roomTempFahrenheit: 104,
  roomPressure: 819,
  ACStatus: 'on',
  personDetector: 'yes',
  lightStatus: 'on',
  roomdoor: 'open',
  timestamp: 2024-05-02T10:59:28.336Z
}
      √ should insert notification messages into the notifications collection based on room data


  2 passing (95ms)
```