



Faculty of Engineering and Applied Science

SOFE 4610U Design & Analysis of IoT Software

Assignment 2

Group #3

Group Member 1

Name: Sarthak Sharma

Student ID: 100604428

Group Member 2

Name: Adam Wong Chew Onn

Student ID: 100598499

Group Member 3

Name: Mitul Patel

Student ID: 100700131

Group Member 4

Name: Matthew English

Student ID: 100704553

10 Step Approach

1.Purpose & Requirement Specification:

Purpose: The purpose of our project is to create a smart cooling system.

Behaviour: The system will be able to sense the temperature and decide whether the fan is turned on or off.

System Management Requirement: The system should provide local monitoring and control functions.

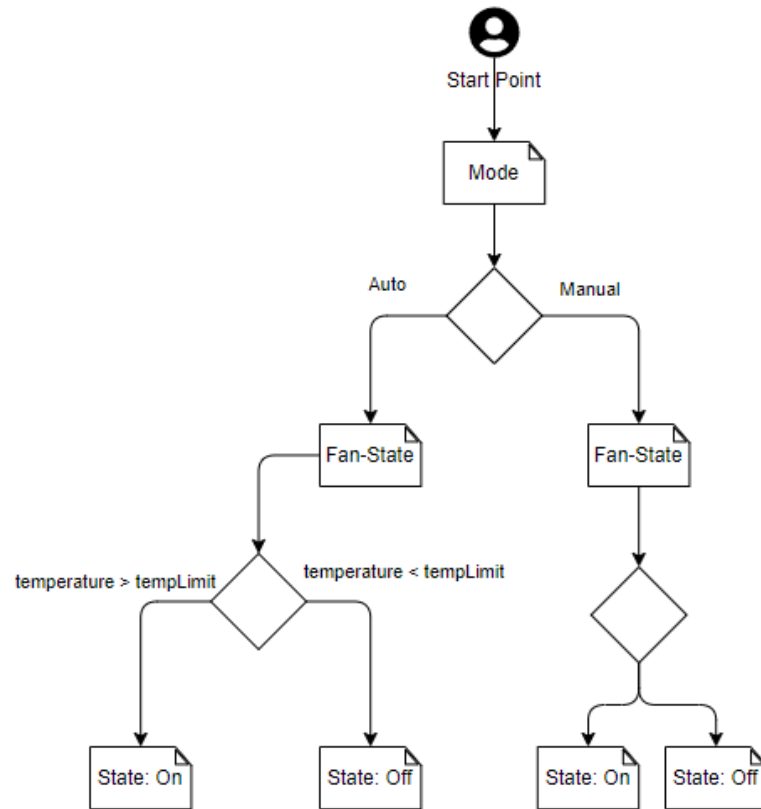
Data Analysis Requirement: The system must read and display the active temperature to the user

Application Deployment Requirement: The application should be deployed locally with a

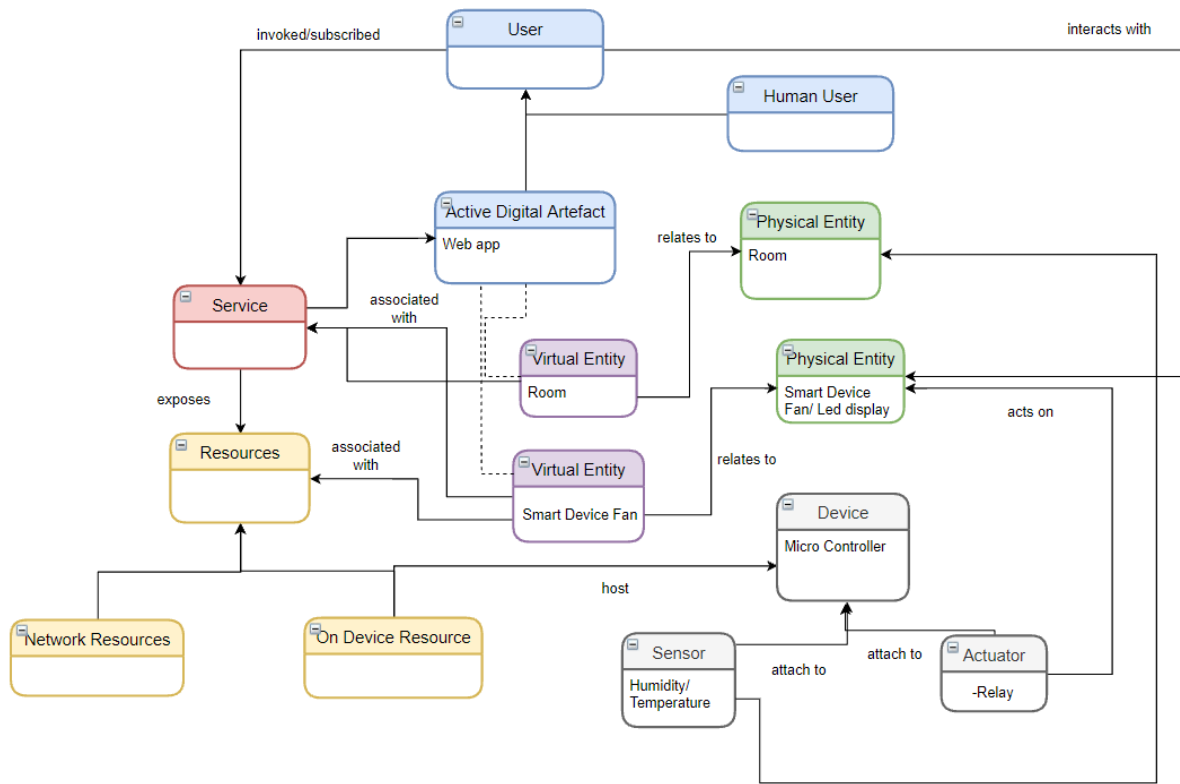
User Interface Requirement: The user should be able to see the current temperature and manually turn the fan on and off

Backend Requirement: The data should be set from the microcontroller to the django backend using a post request. Furthermore, the application should display temperature data on the app.

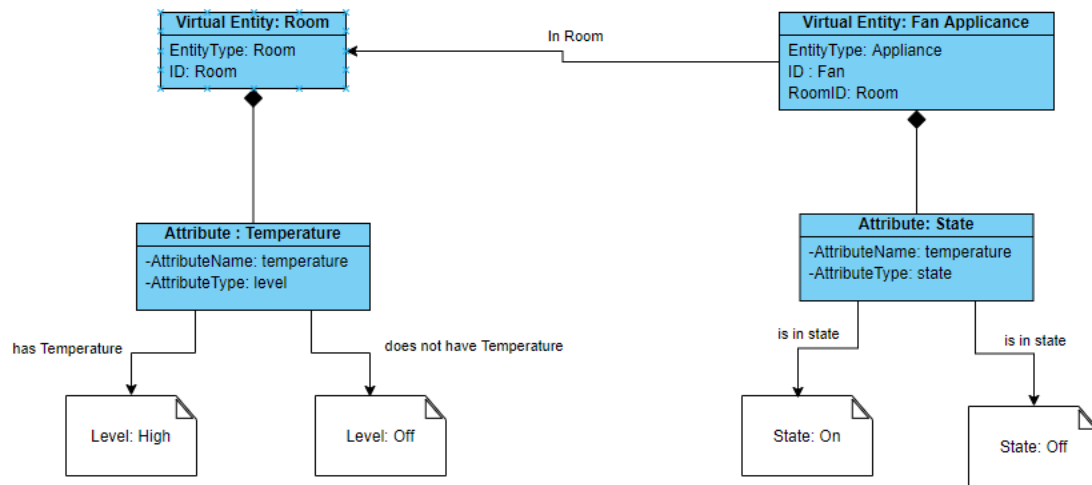
2.Process Specification:



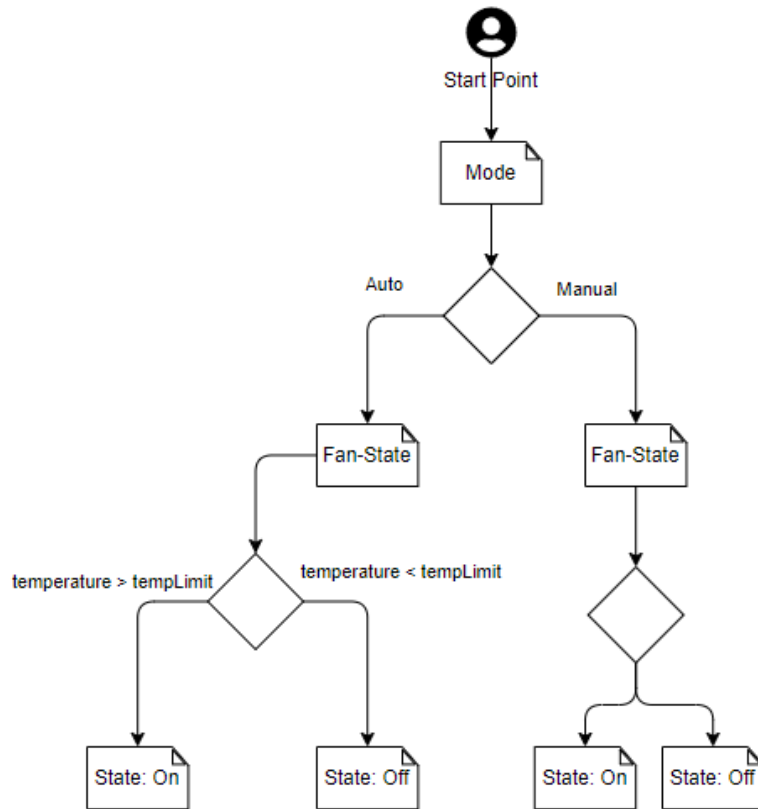
3.Domain Model Specification:



4.Information Model Specification

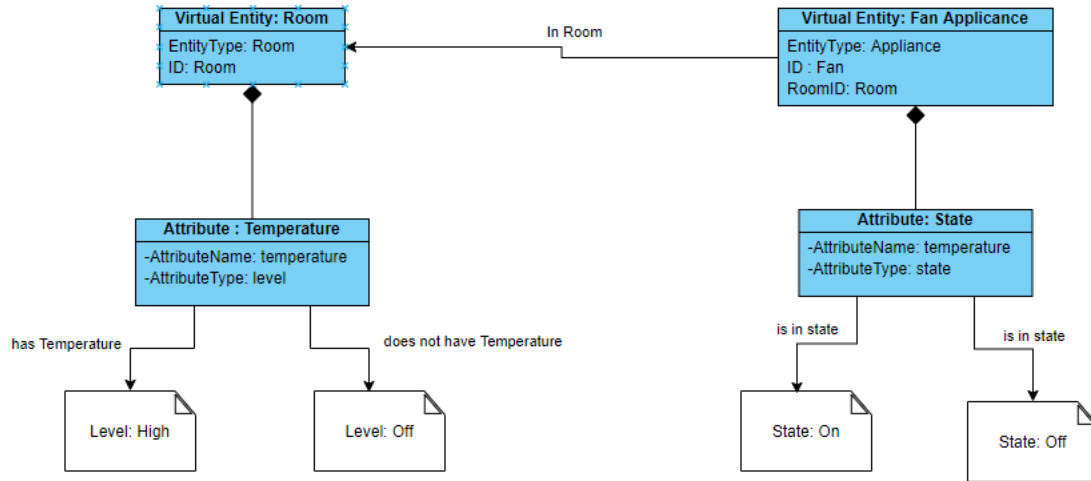


5.Service Specifications:



Mode Service allows the user to manually turn the fan on and off instead of the auto mode.

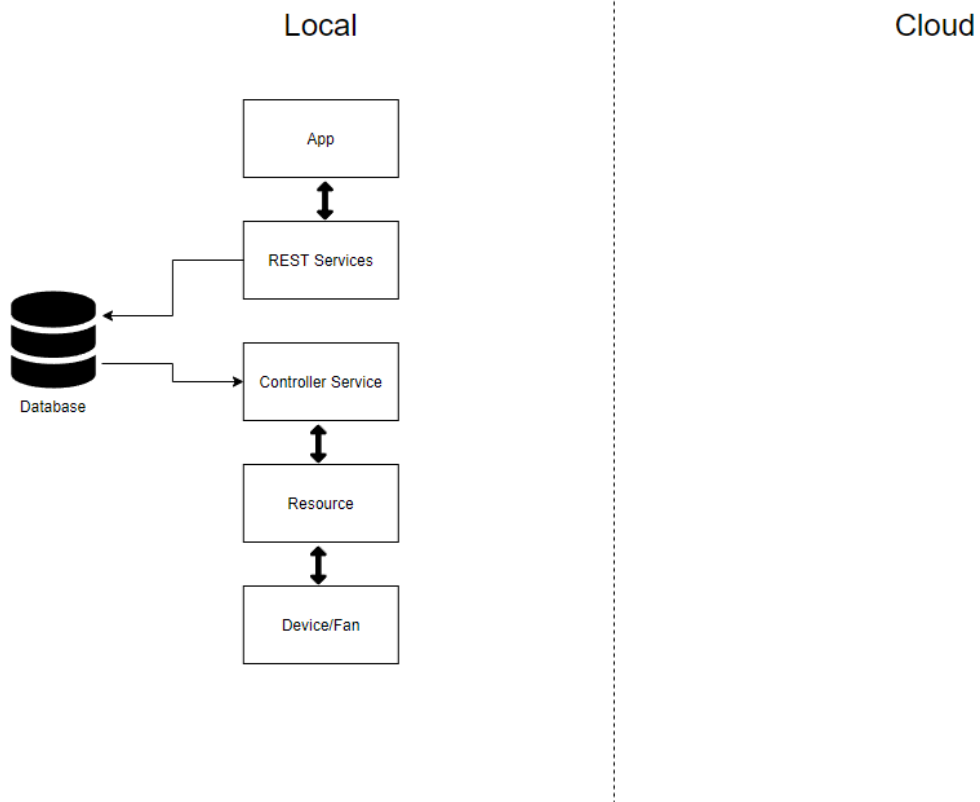
State Service: Sets the fan appliance on and off depending on current temperature.



State Service: Sets the fan appliance on and off depending on current temperature.

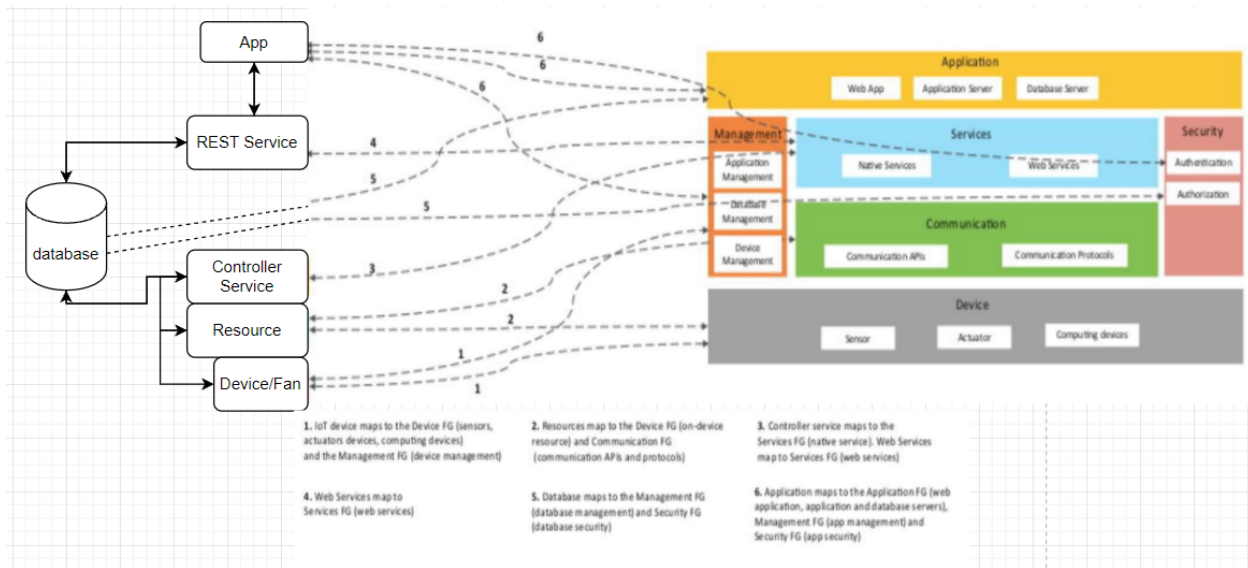
Controller Service: Auto mode monitors the temperature and switches the fan on and off. In manual mode, the user is able to change the state of the fan by switching it on/off.

6.Iot Level Specification:



7.Functional View Specification:

Diagram



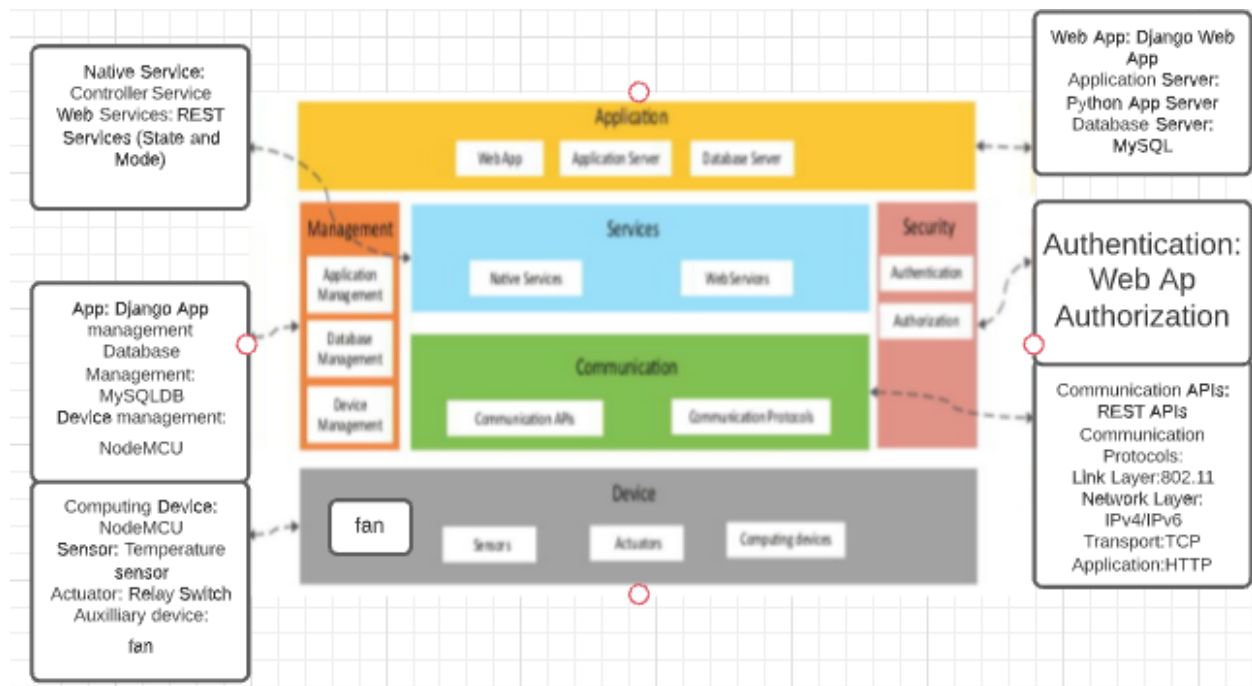
User case:

- UC 1-** User should be able to view their device information from the application
- UC 2-** Devices must communicate with Django applications with high level api rest calls.
- UC 3-** Users should be authenticated using tokens and app should manage users
- UC 4-** The device should turn on the fan when the temperature passes a certain range.

These use cases are mapped to the application developed below. The mappings are available on the github along with the project: https://github.com/Mitul2000/Smart_Cooling_System_IoT

8.Operational View Specification:

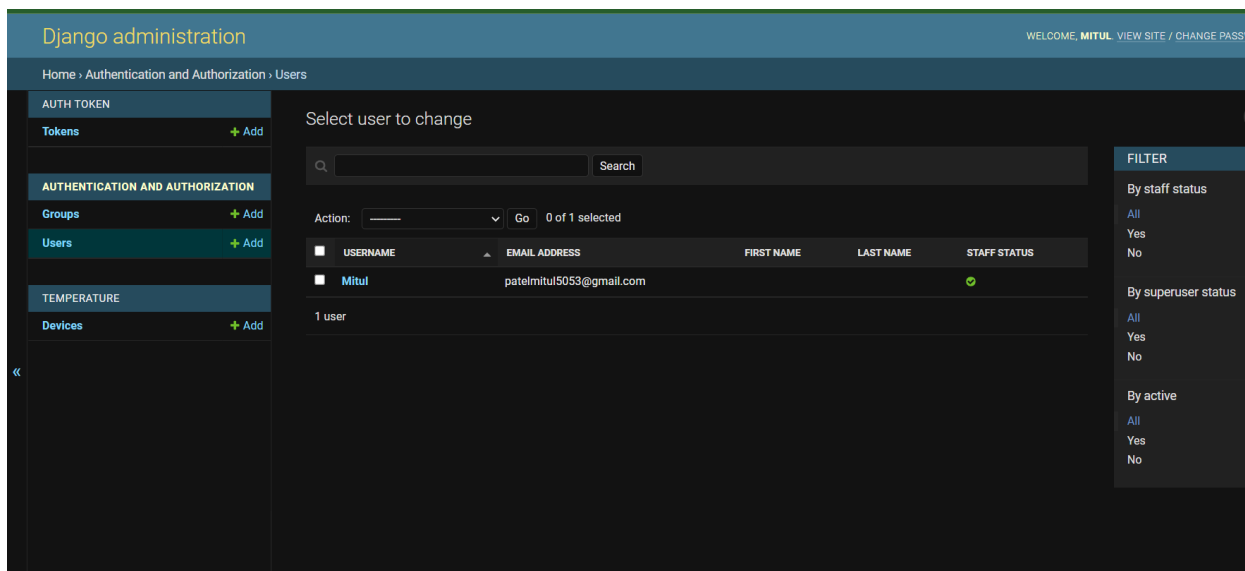
Diagram



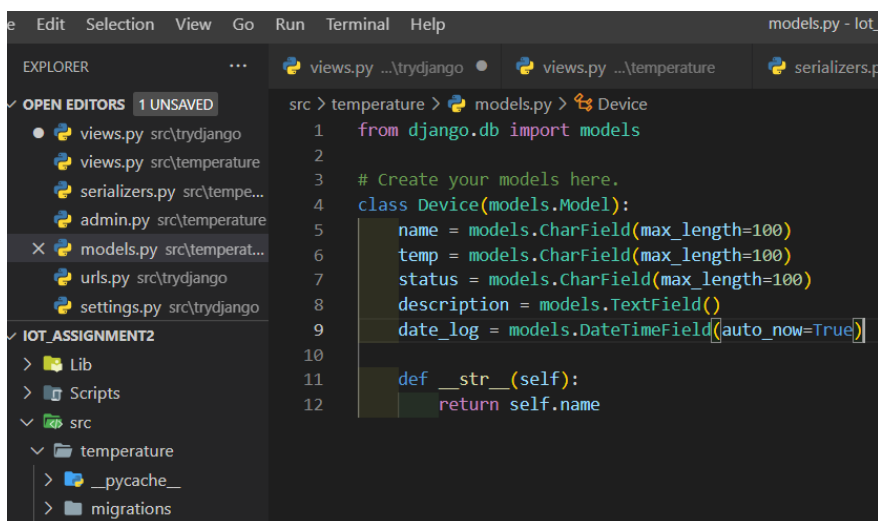
2. Implement your design using the Django REST framework. Submit a GitHub link of the code and tests for the implementation. In the README file create a table of contents to help navigate your code and show that your implementation performs the functionality if the use case is picked for design.

Github link : https://github.com/Mitul2000/Smart_Cooling_System_IoT

Admin page:



Device Model



User Token:

Home > Auth Token > Tokens

AUTH TOKEN

Tokens [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

Select token to change

KEY	USER	CREATED
76a37de20178f7b898b748efc4c66908eca64e78	Mitul	Oct. 31, 2021, 9:39 p.m.

1 token

Test 1:

Making POST Request with authentication token:

http://127.0.0.1:8000/ [Save](#) [Edit](#)

POST [Send](#) http://127.0.0.1:8000/ [Cookies](#)

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

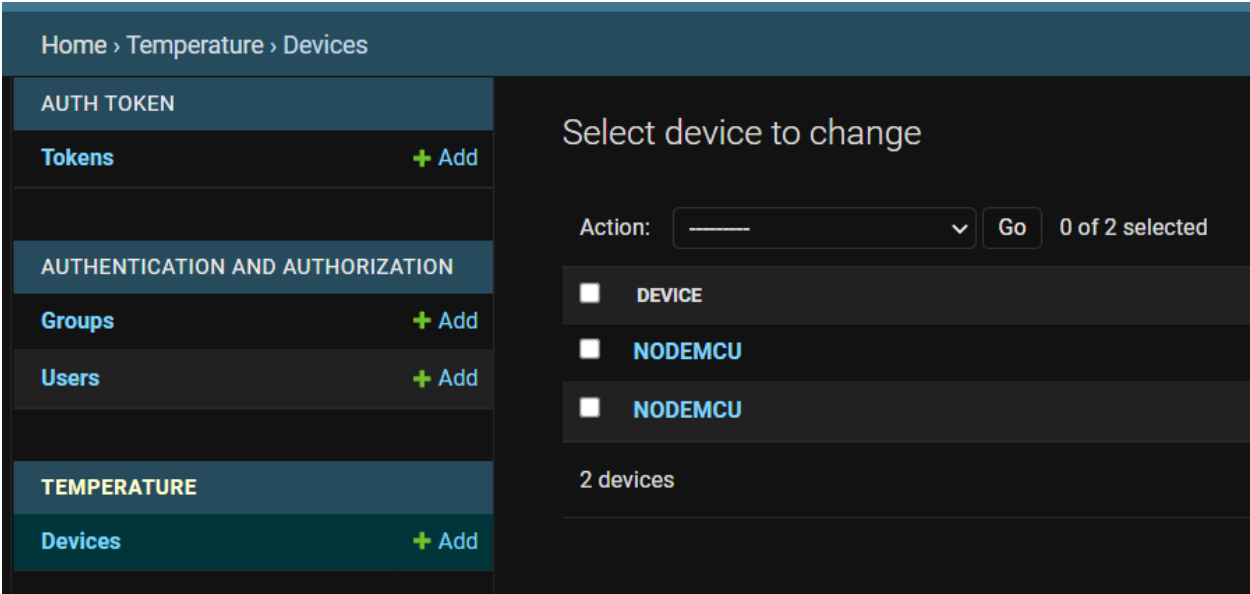
	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	NODEMCU			
<input checked="" type="checkbox"/>	temp	28.4 C			
	Key	Value	Description		

Body Cookies Headers (9) Test Results [Save Response](#) Status: 200 OK Time: 30 ms Size: 313 B

Pretty **Raw** Preview Visualize

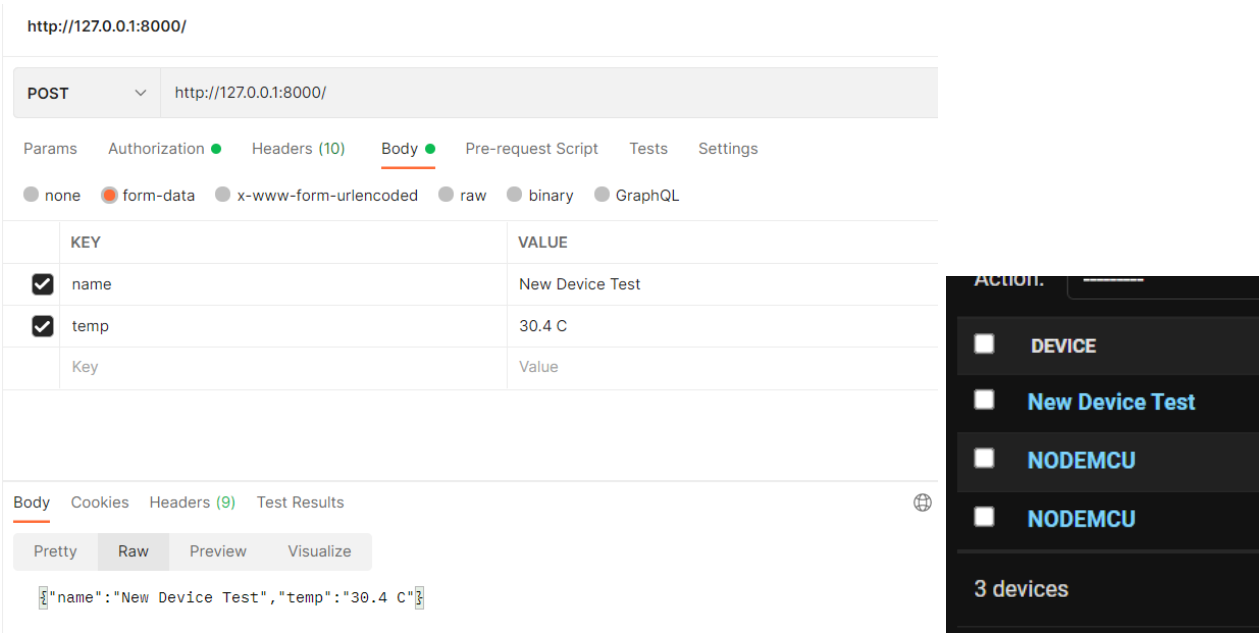
```
[{"name": "NODEMCU", "temp": "28.4 C"}]
```

Device Object in admin view



Test2:

Making POST Request with authentication token for registering new device:



New Device attributes:

Django administration

Home > Temperature > Devices > New Device Test

AUTH TOKEN

Tokens

+ Add

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

Users

+ Add

TEMPERATURE

Devices

+ Add

Change device

New Device Test

Name:

New Device Test

Temp:

30.4 C

Status:

Description:

Delete

Test 3: sending post request with authentication token

POST http://127.0.0.1:8000/

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	name	Auth Test
<input checked="" type="checkbox"/>	temp	30.4 C
	Key	Value

Body

Cookies

Headers (10)

Test Results

Pretty

Raw

Preview

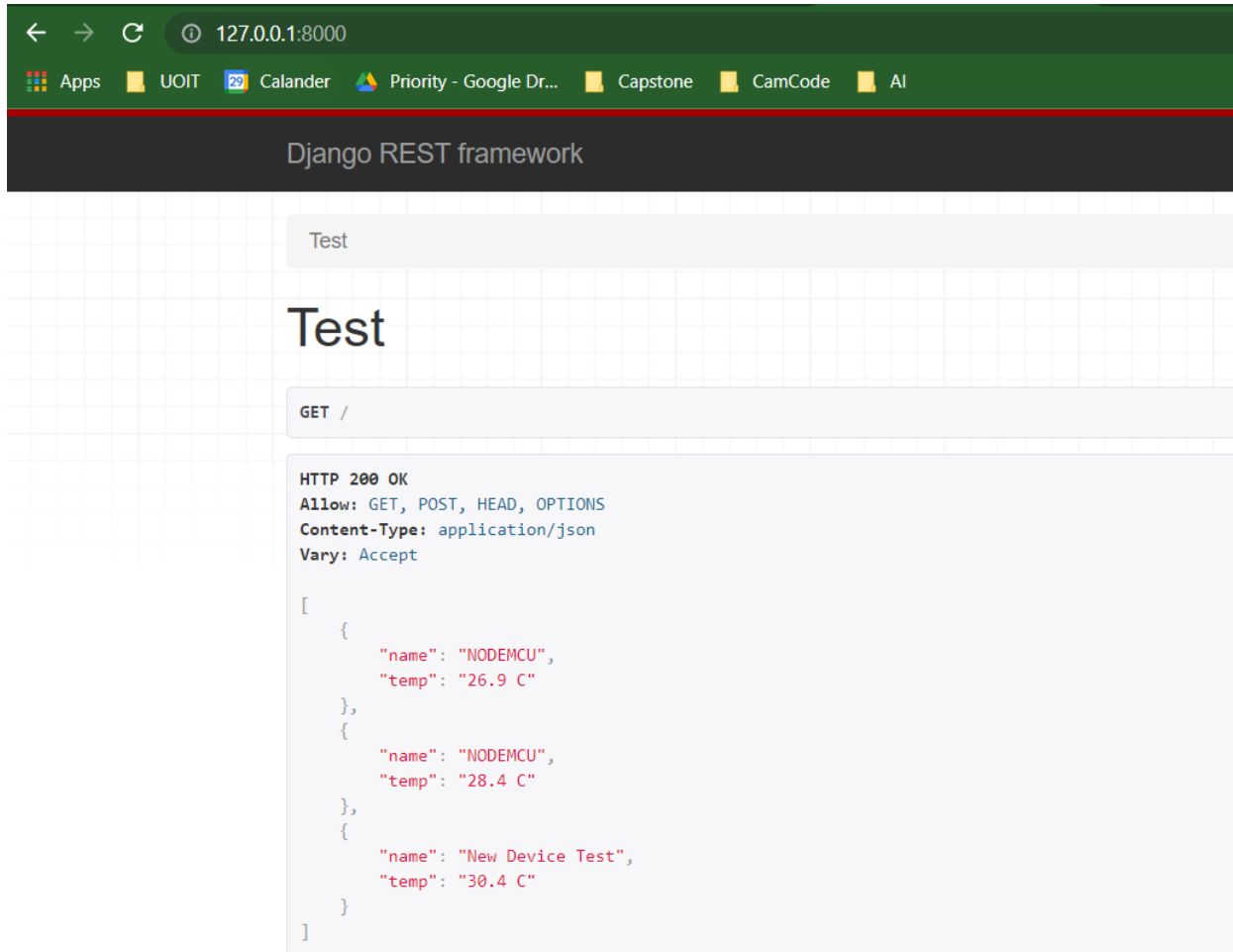
Visualize

"detail": "Invalid token."

Test 4:

Get request to view all devices and temperature on dashboard page:

This list can also be filtered based on the user.



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000`. The browser's tab bar includes several tabs: "Apps", "UOIT", "Calander", "Priority - Google Dr...", "Capstone", "CamCode", and "AI". The page content is a Django REST framework interface. At the top, it says "Django REST framework". Below that, there's a "Test" label. The main heading is "Test". Underneath, it shows the request method "GET /". The response is "HTTP 200 OK" with headers: "Allow: GET, POST, HEAD, OPTIONS", "Content-Type: application/json", and "Vary: Accept". The response body is a JSON array of three objects, each representing a device with its name and temperature.

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
  {
    "name": "NODEMCU",
    "temp": "26.9 C"
  },
  {
    "name": "NODEMCU",
    "temp": "28.4 C"
  },
  {
    "name": "New Device Test",
    "temp": "30.4 C"
  }
]
```