**TESTING SCREENSHOTS**

**Using Swagger, Curl and Postman**

Positive scenarios:

**This covers all the end-points shown via swagger (as given below)**

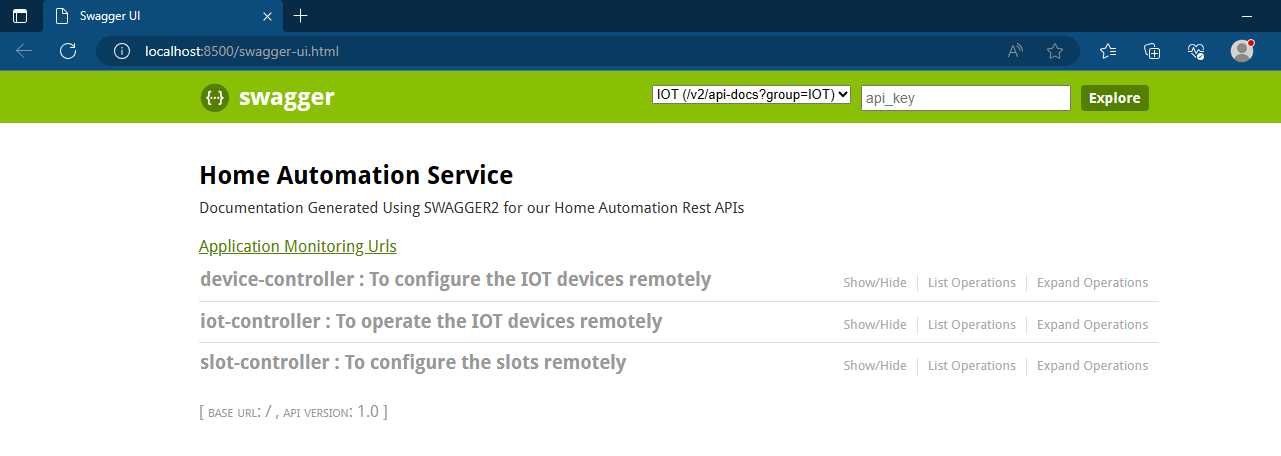
The “Home Page” comprises of 3 controllers

a) Slot Controller 🡪 *SlotController is used to configure the slots remotely*

b) Device Controller 🡪 *DeviceController is used to configure the IOT devices remotely*

c) Iot Controller 🡪 *IotController is used to operate the IOT devices remotely*

The testing is via Swagger and/or Postman and/or even using CURL commands. The steps are briefly captured while retrieving the test results

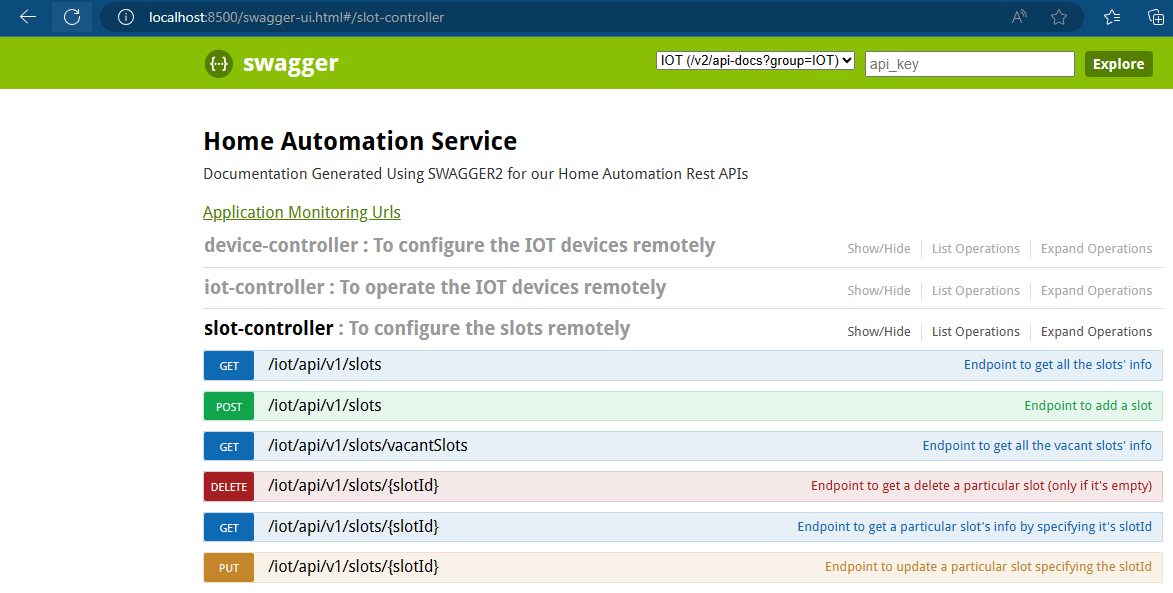


**TESTING USING SWAGGER**

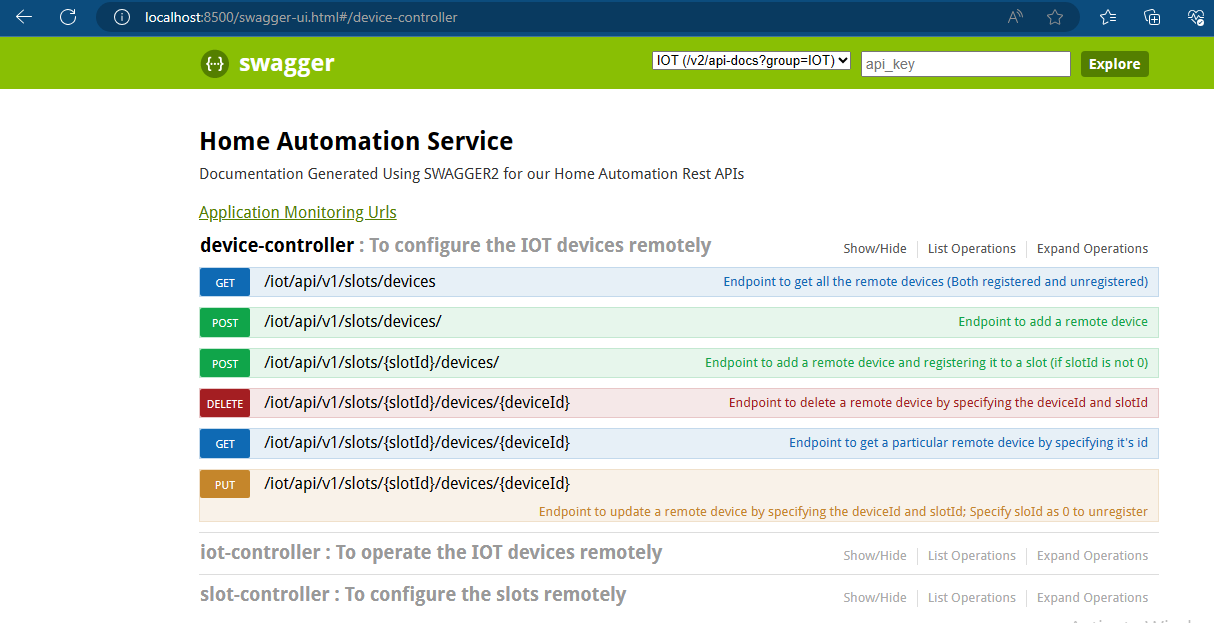
Endpoints can be viewed by clicking on the corresponding controller(s)

We need to create a slot using slot controller and then device using device controller and finally perform the operations, once device is associated with a slot

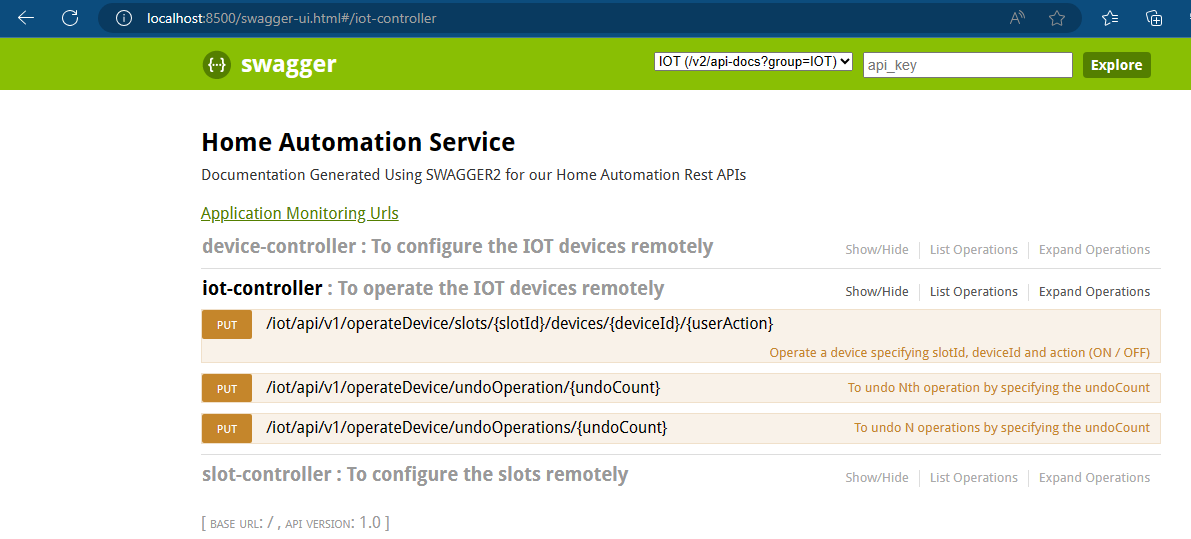
Endpoints of slot-controller



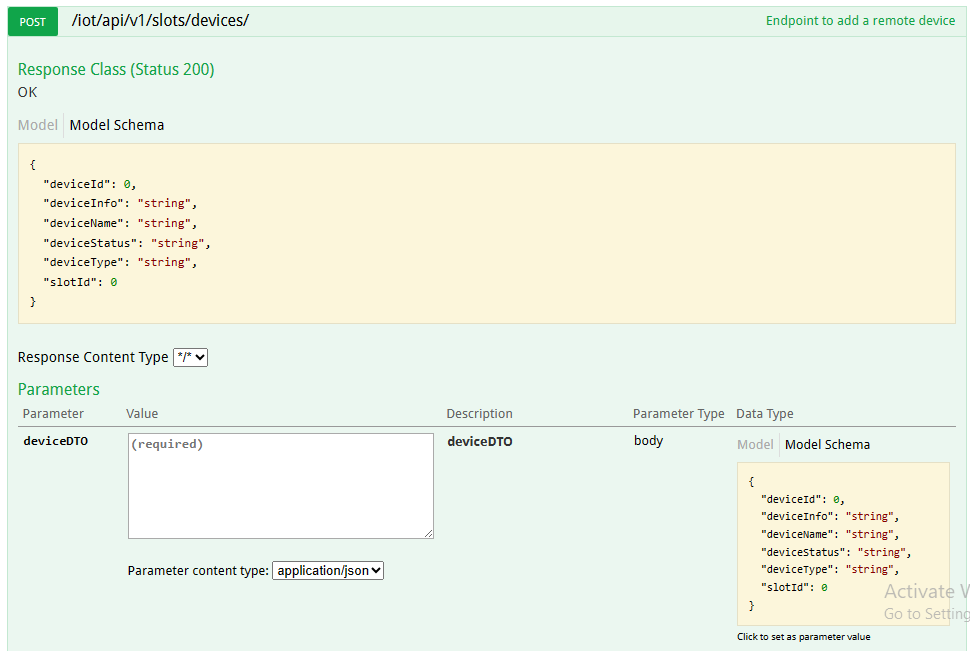
Endpoints of device-controller



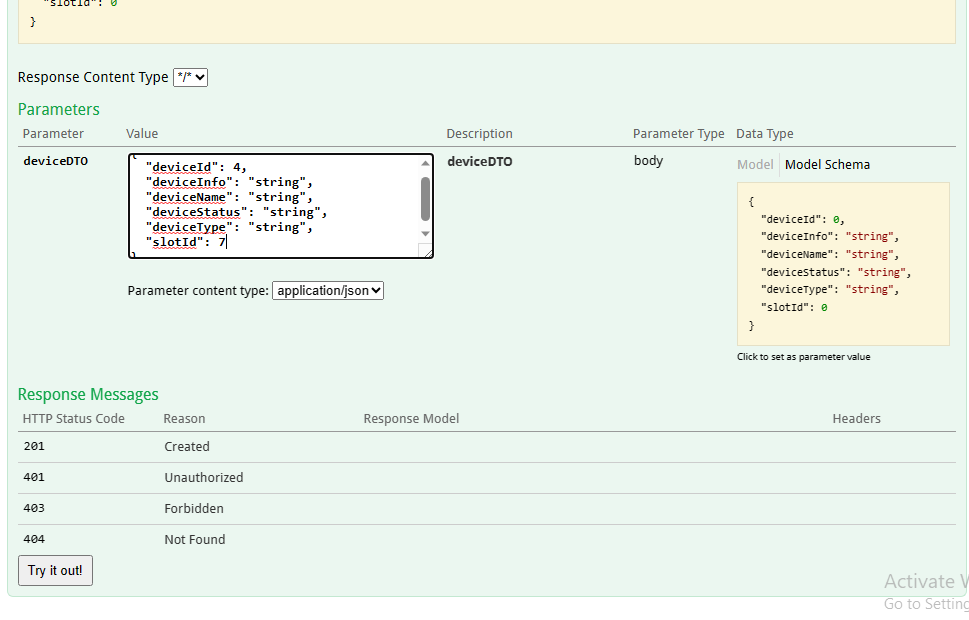
Endpoints of iot-controller



The payload details can be seen by clicking on the respective endpoints



Also testing can be done by keying in the values in all fields mentioned as “required” and clicking on the button “Try it out”



The response details are populated as shown below.

It shows

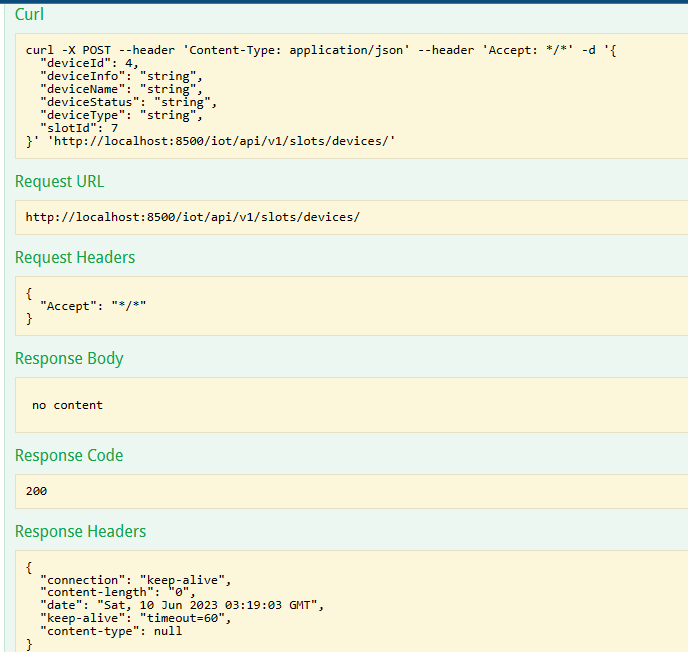
# Request URL,

# Response header,

# Response code

# Response body.

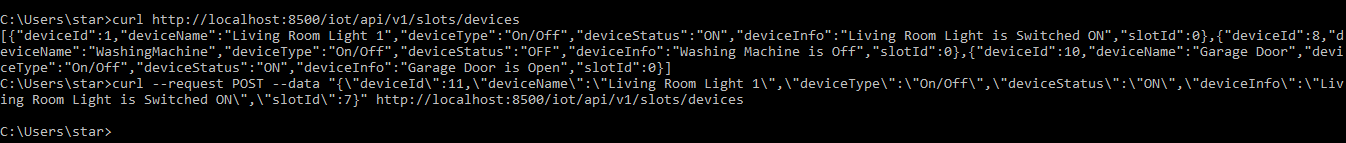
Also additionally it gives the corresponding CURL commands that swagger internally uses, for execution.



**TESTING USING CURL**

Endpoints can be viewed by running the CURL commands from command line

Sample testing using CURL commands



**TESTING USING POSTMAN**

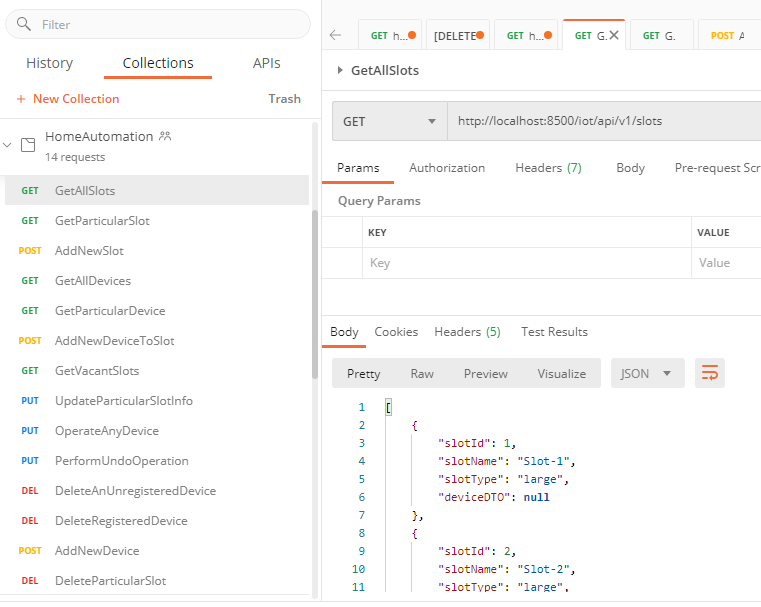
Endpoints can be tested via postman too by going to

<https://documenter.getpostman.com/view/10383827/2s93sabtT5>

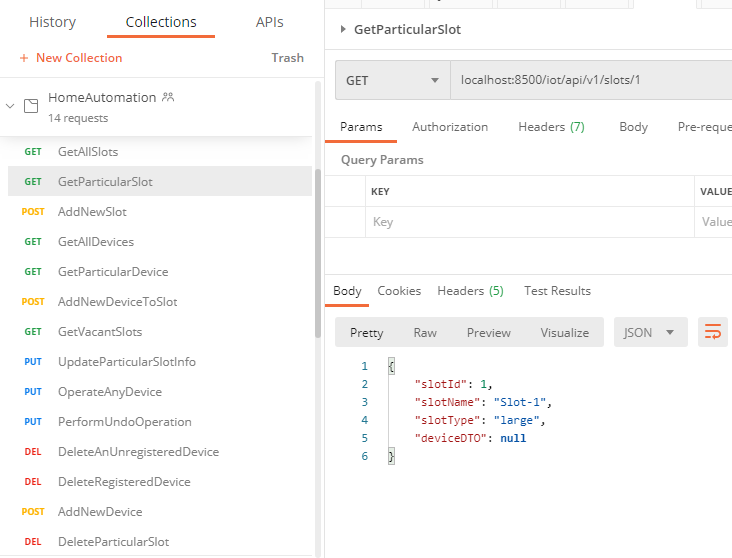
The collection URLs are specified in the path

<https://github.com/deepak2all/HomeAutomation/blob/master/documents/HomeAutomation.postman_collection.json>

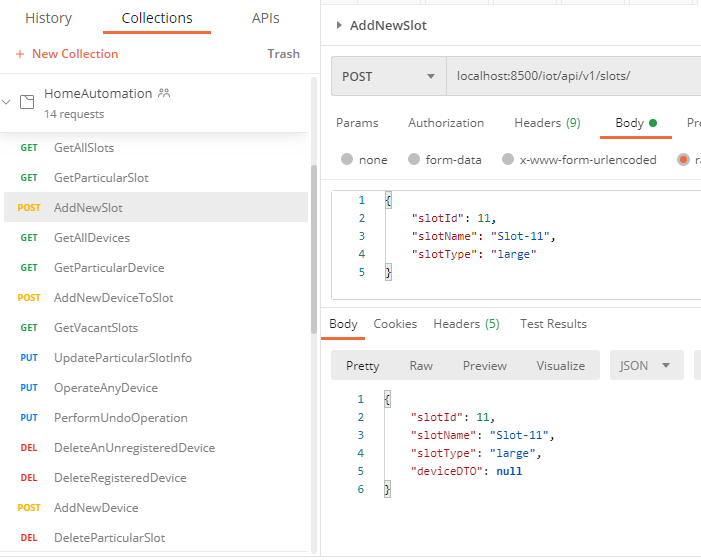
**1. Get All Slots**



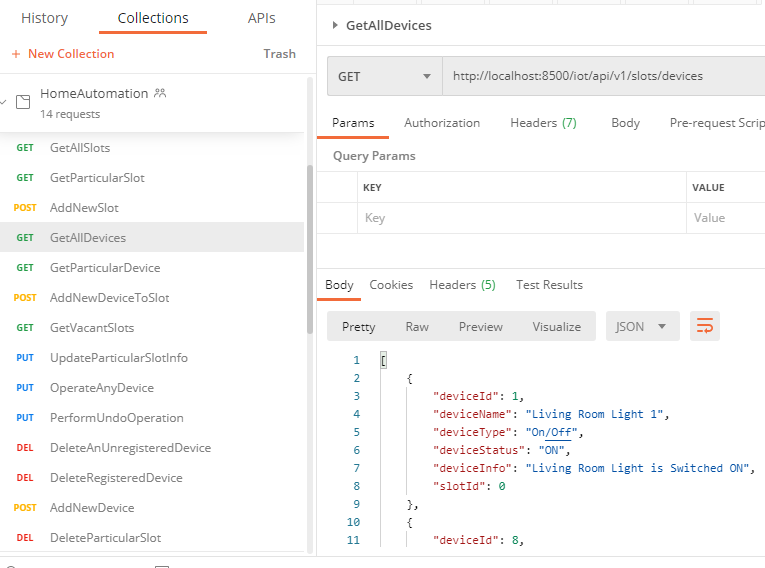
**2. Get Particular Slot**



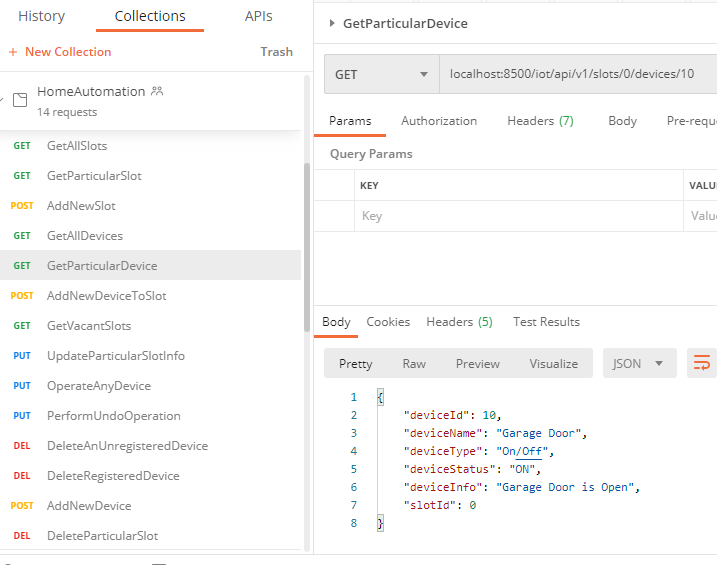
**3. Create New Slot**



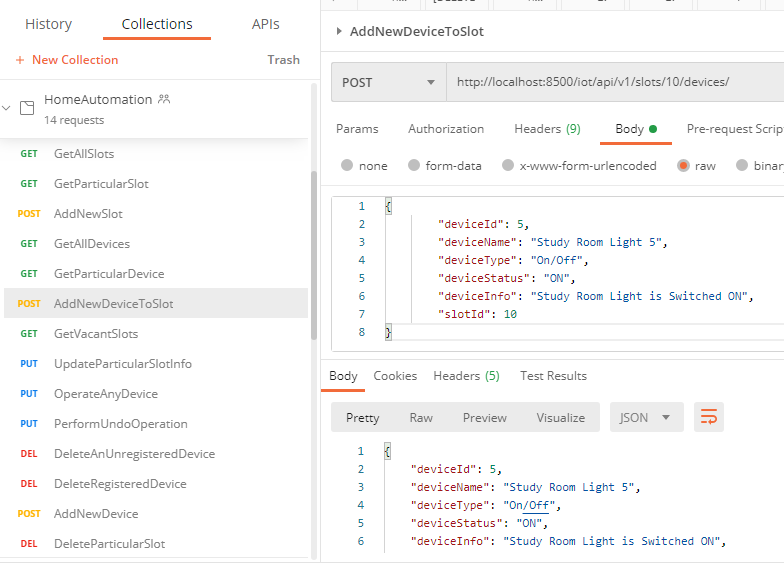
**4. Get All Devices**



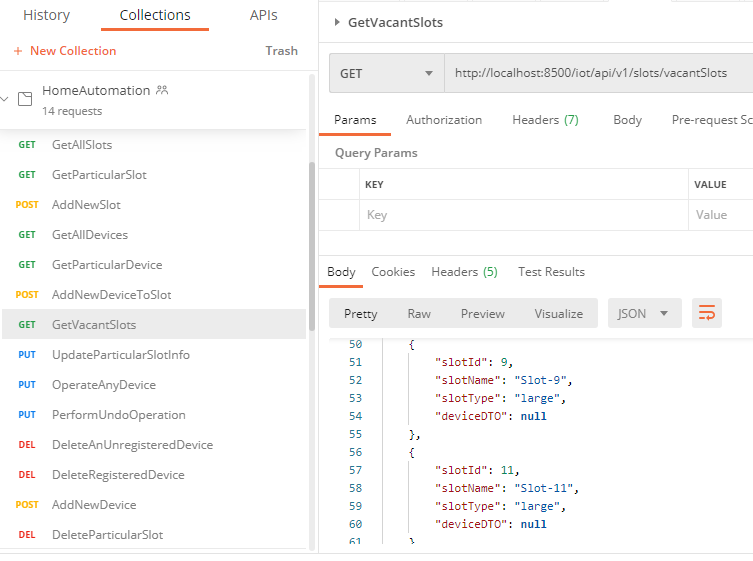
**5. Get Particular Device**



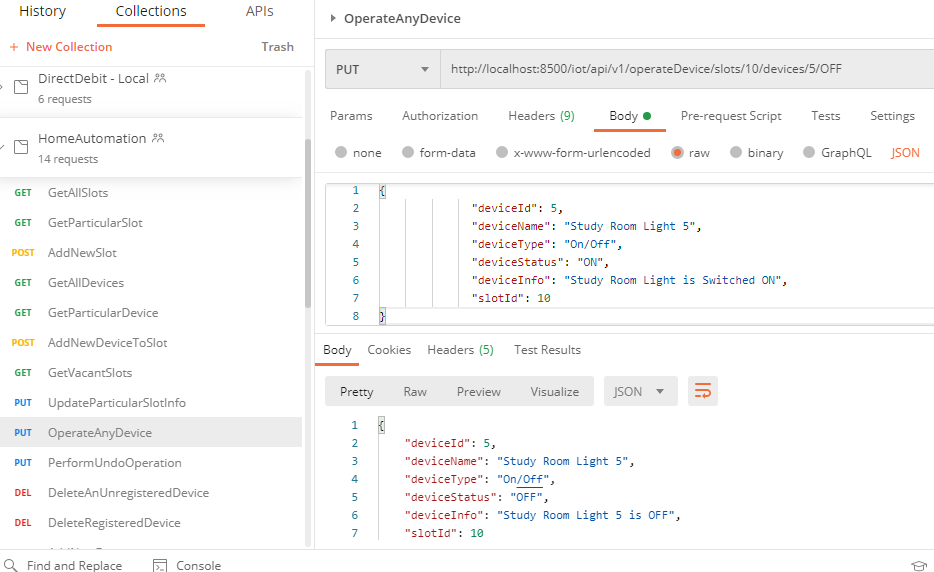
**6. Add New / Unregistered Device to Slot**



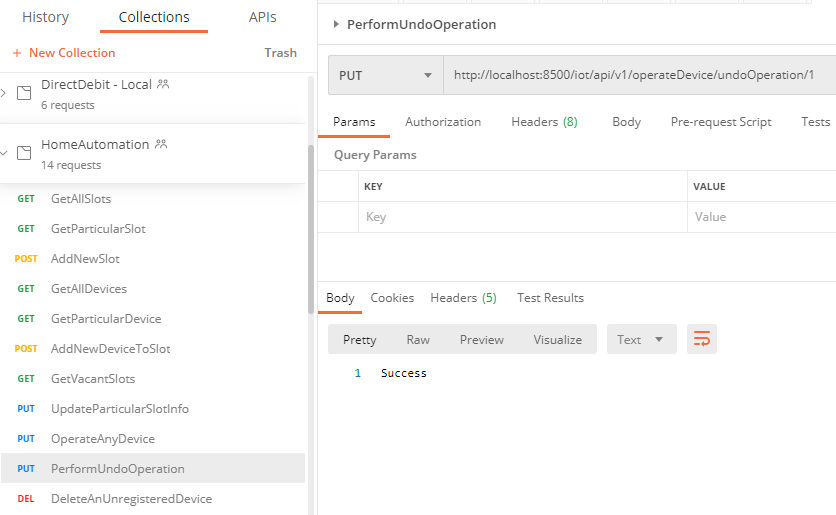
**7. Get Vacant Plots**



**8. Operate any device**



**9. Perform Undo Operation**



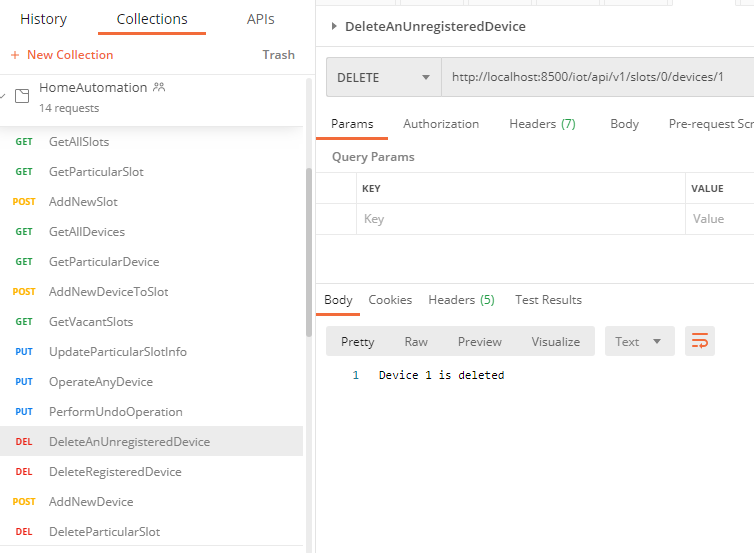
Imagine the device is in ON state and added to the slot (This action is not captured as it pertains to adding of the device)

Subsequently the state of the device is changed to OFF, in 1st operation

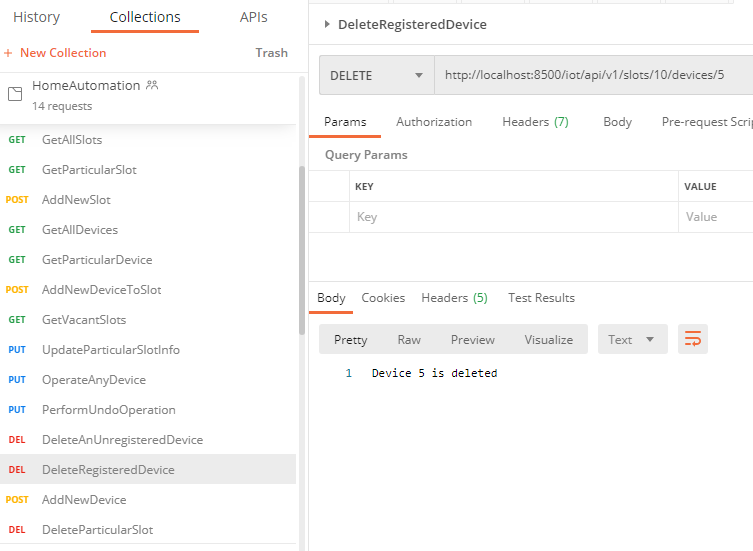
Then the state of the same device is changed to ON, in 2nd operation

Now, with the undo operation, that state of the device will return back to OFF

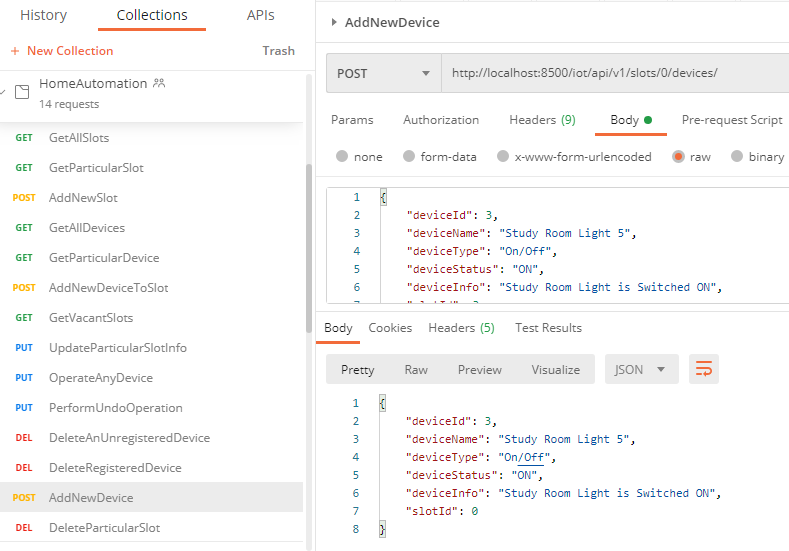
**10. Delete an unregistered device**



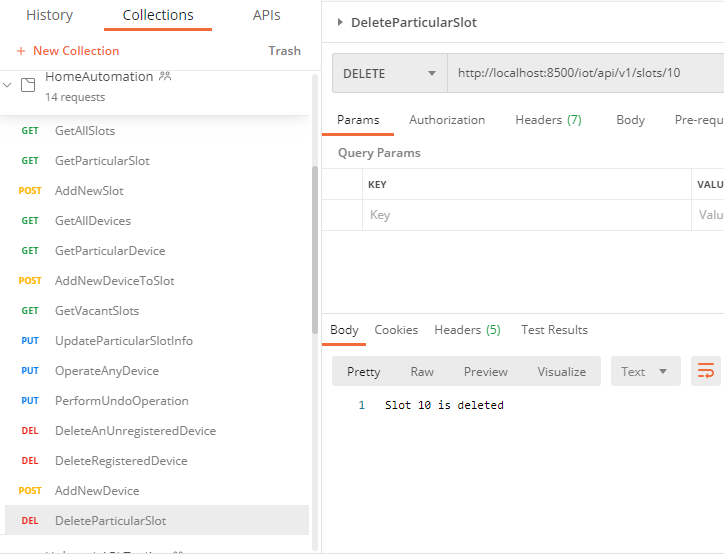
**15. Delete a Registered device**



**16. Add a new device**



**17. Delete particular slot**



Negative scenarios:

**This section covers the key –ve scenarios**

**1) Device / Slot as -ve numbers**

Expected Result: Throw an exception stating incorrect input provided

Actual Result: Matches expected

**2) Device / Slot as strings**

Expected Result: Throw an exception stating incorrect input provided

Actual Result: Matches expected

**3) Re-creating a new slot with same slotId / deviceId (pre-existing)**

Expected Result: Newer value to be updated (can introduce a constraint when the data is added to DB)

Actual Result: Matches expected

**4) Status other than ON / OFF**

Expected Result: Throw an exception asking to "Specify the device state as 'ON' / 'OFF'"

Actual Result: Matches expected

**5) Associating device with slot that doesn't exist**

Expected Result: Throw an exception stating incorrect slot is provided

Actual Result: Matches expected

**6) Associating a non-existent device to a slot**

Expected Result: The new device will be created and associated to respective slot

Actual Result: Matches expected

**7) Associating a non-existent device to a slot that doesn't exist**

Expected Result: Throw an exception stating incorrect slot is provided

Actual Result: Matches expected

**8) Associating a device to a slot which is already associated with another device**

Expected Result: Throw an exception stating incorrect slot is provided

Actual Result: Matches expected

**9) Operating a device not inserted in a slot**

Expected Result: Throws exception stating “The slotId / deviceID in the path / body cannot 0“

Actual Result: Matches expected

**10) Operating a non-existent device**

Expected Result: Throws exception stating “Incorrect slotId / deviceID specified“

Actual Result: Matches expected

**11) Operating a device from incorrect slot**

Expected Result: Throws exception stating “Incorrect slotId / deviceID specified”

Actual Result: Matches expected

**12) Operating a device by keying incorrect status**

Expected Result: Throws exception stating “Specify the device state as 'ON' / 'OFF'”

Actual Result: Matches expected

