FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Mobile Computing Project WS-19/20

Location Based Service-IOT Smart Home Application

Under *Guidance of*:

- **Prof. Armin Lehmann**
- **Mr. Gregor Frick**
- **Mr. Besfort Shala**

Submitted by

First Name: Abhishek

Last Name: Sharma

Matriculation No. :- 1274291

Email : asharma@stud.fra-uas.de

Date: 16.01.2020

Course: Masters in Information Technology

Department of Telecommunication Networks

Frankfurt University of Applied Sciences

60318 Frankfurt/Main, Germany

Table of Contents

Acknowledgement

I would like to thank Prof. Lehmann, Mr. Frick , Mr. Shala for the continuous support and guidance provided for this project. I am very much thankful for providing the open platform via moodle to ask question , clearing doubts and sharing valuable suggestions.

Furthermore, I would also like to put my sincere gratitude for providing all the necessary information and useful links via moodle which proved to be very helpful for successful completion of this project.

Abstract

The growing use of IOT(Internet of things) in Location-based service provides an opportunity to find solutions for problems and challenges in the rapidly changing telecommunication networks and become important part of Internet of things where location of everything (LOE) plays an important role in making various smart home application in IOT area.

The purpose of this document is to demonstrate the development and implementation of Location based service -a smart home web application following Http Rest web service and CoAP (IOT) Protocol in an Emulated IOT Environment using Java Technologies. The technologies include Java spring-boot framework, HTML5(Bootstrap), XML and JSON data binding. The developed components are Location sensors, Actuators, IOT Gateway, web client. The Use-cases of this project are designed with respect to Smart-Home Automation as per project guidelines.

This complete project is a part of Mobile Computing module of Frankfurt university of applied science. In this document I will discuss about the Principle of project ,Design architecture, Project Implementation & Application logic, Use-cases, Manual to operate, diagrams and implementation of Http and CoAP protocol.

1.Introduction

IOT applications relying on position have become part of everyday life leading to an increasing variety of location-based services in various domain such as Smart-home, Smart-city, Smart-car etc. This project is based on the Smart-home use case which I will discuss in detail in subsequent section.

1.1Principle and Architecture of Project

The principle of this project is based on client- server and Request-Response model using Rest service ,Http protocol , CoAP(IOT) Protocol and IOT Gateway.

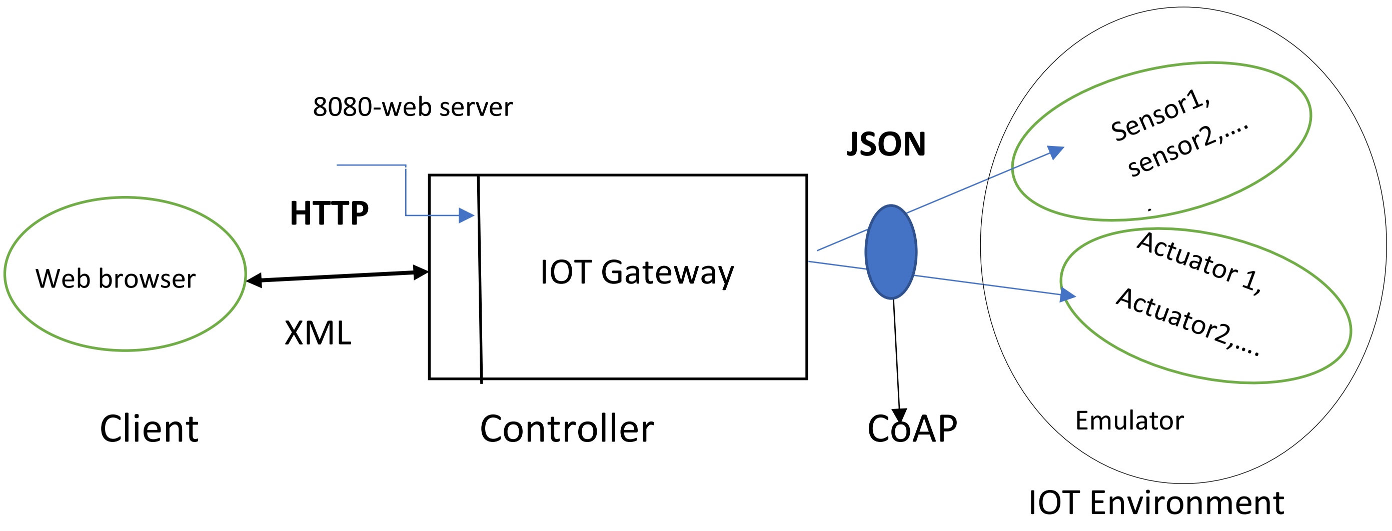


Fig :1 General Network Architecture

1. Client :- web browser is acting as a client which is responsible for sending Http Request and Receiving Http Response in XML Data Format. REST(Representational State Transfer) web service is created in order to handle GET and POST Http Request Response.

2. Controller:-Controller is act as IOT Gateway. It is responsible for Protocol translation from Http to coAP and is responsible for handling Request- Response from web browser to sensor actuator along with the required xml &JSON Format conversion. Controller one end is act as a web server and other end act as coap client therefore, called as Gateway.

3. IOT Emulated Environment:- This environment is created while defining sensors and actuator as a java application(Fig: 1). All sensors and actuators are acting as separate resource running on different coAp ports.

1.2 USE-CASE of Project

I have define my use case of smart home using Location based service in three different scenario:

1. **Pet(DOG) safety:-** In general, Considered the scenario where owner of the house wants to make sure the safety of its Dog using Location service(IOT) of smart home.
   Dog is allowed to roam freely in the big house yard (let say, 100 meter) ,while doing so there is possibility of escaping of dog from the Main Gate which is usually semi-opens .Therefore, smart home is developed to protect dog escaping from Main Gate and even in case, dog jumps from wall or Gate(try to go beyond 100m),Alarm must be trigger in the house alerting the owner(user).
   In short, when dog comes near main gate(let say 70 meter), Automatically Gate will close and even if dog somehow jumps from the Gate ,Automatically Alarm will trigger.

2. **Power saving via Lights and ventilation via Window:-** This is a scenario when house owner(treated as user here) comes nearby home Light will on Automatically and when user move away from home Light will off automatically. In similar manner, Window opens at user nearby home location automatically and closes at user away from home automatically.

3. **Kids Safety :**Considering the scenario when kids play nearby home say park but needs a continuous monitoring by parents. Parents wants to develop IOT application in order to track kids location(during playing) nearby home and intentionally defined kids playing location range where kids are not allowed to go.

   Here, in this project, First I will set the Kids location beyond which if kids move I will get an Alert. When kids are playing under defined location that is treated as safe zone and when kids try to go beyond defined location -an alert message will trigger**.**

Note: This is a Non-technical definition of use case , I will discuss technical approach of use case along with all sensors , actuators and different parameter under Project Implementation.

2.**CoAP Protocol** [RFC 7252]

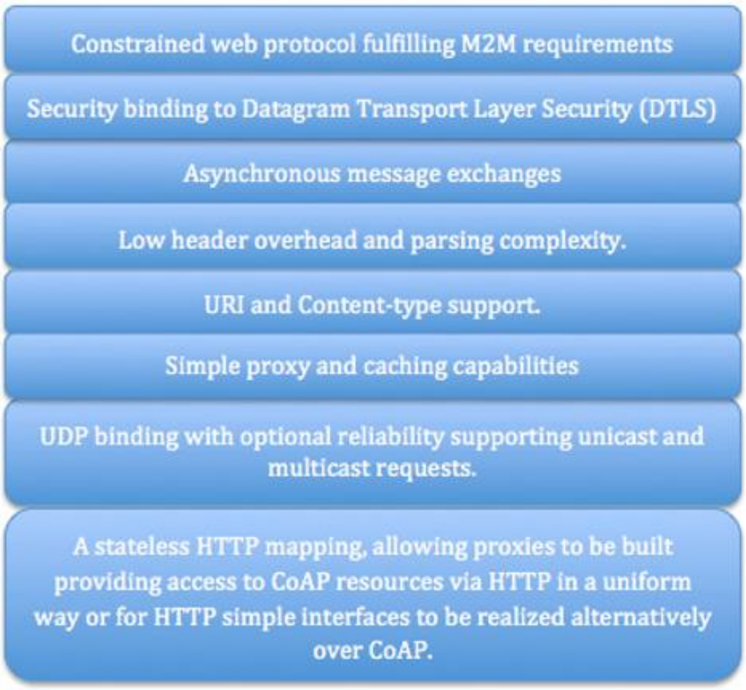
The use of web services for sensor networking applications is seen as an important part in emerging M2M and IOT communications. The Constrained Application Protocol (CoAP) is proposed by the IETF(open standard RFC 7252) to optimize the use of the RESTful web service architecture in constrained nodes and networks, for example Wireless Smart-Home Networks.

CoAP is one of the latest application layer protocol developed by IETF for smart devices to connect to Internet. Thus lightweight protocol CoAP is intended to be used and considered as a replacement of HTTP for being an IoT application layer protocol[2].

Unlike HTTP based protocols, CoAP operates over UDP and employs a simple retransmission mechanism instead of using complex congestion control as used in standard TCP. It uses a unique Transaction ID to identify each GET request for retransmission purposes to keep reliability.

2.1 CoAP Features

CoAP needs to consider optimizing length of datagram and satisfying REST protocol to support URI (Uniform Resource Identifier). It also needs to provide dependable communication based on UDP protocol[5].



Constrained web protocol fulfilling M2M requirements
Security binding to Datagram Transport Layer Security (DTLS)
Asynchronous message exchanges
Low header overhead and parsing complexity.
URI and Content-type support.
Simple proxy and caching capabilities
UDP binding with optional reliability supporting unicast and multicast requests.
A stateless HTTP mapping, allowing proxies to be built providing access to CoAP resources via HTTP in a uniform way or for HTTP simple interfaces to be realized alternatively over CoAP.

Table 1 : CoAP Feature

2.2 CoAP Structure Model

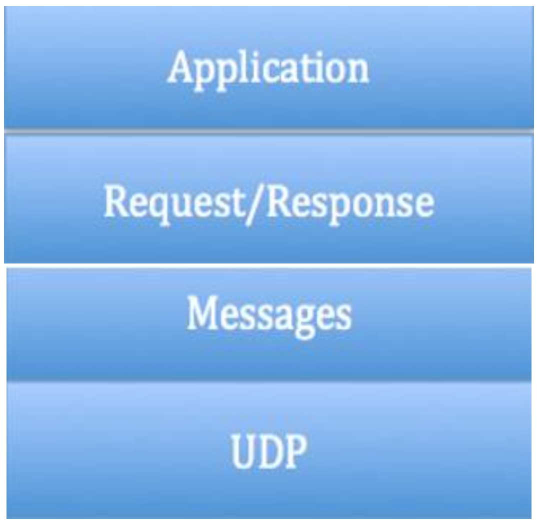


Fig:2 Abstract Layer of CoAP

CoAP employs a two layers structure. The bottom layer is **Message layer** that has been designed to deal with UDP and asynchronous switching. The **request/response** layer concerns communication method and deal with request/response message.

2.2.1 Message Layer Model

Message Layer supports 4 types message: **CON (confirmable)**, **NON (non-confirmable)**, **ACK (Acknowledgement), RST (Reset)**

a) Reliable message transport: Keep retransmission until get ACK with the same message ID (like 0x8c56 in fig. 2). Using default time out and decreasing counting time exponentially when transmitting CON. If recipient fail to process message, it responses by replacing ACK with RST. Fig. 2 shows a reliable message transport[6].

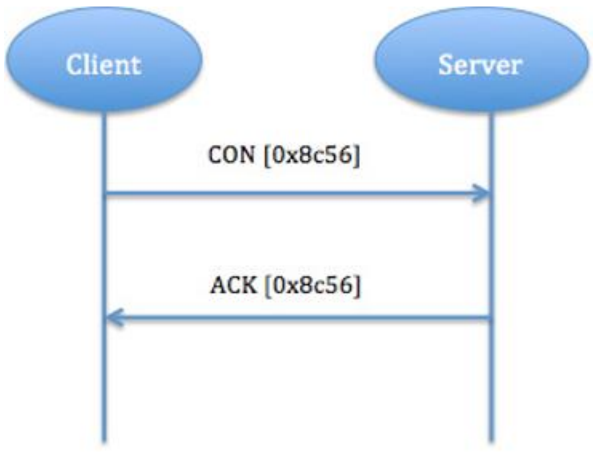
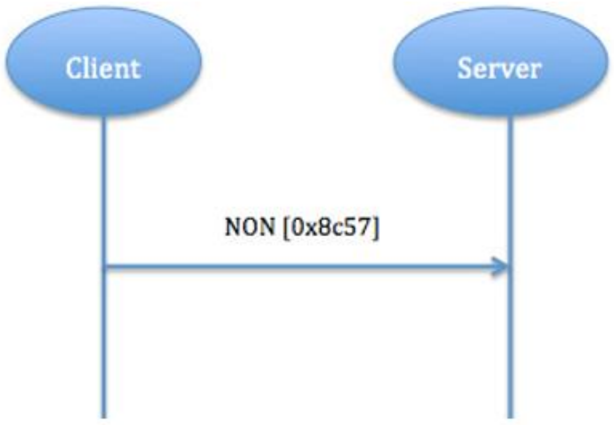


Fig:3 Reliable Message Transport

b) Unreliable message transport: transporting with NON type message. It doesn't need to be ACK, but has to contain message ID for supervising in case of retransmission. If recipient fail to process message, server replies RST. Fig. 3 shows unreliable message transport.



. Fig: 4 Unreliable Message Transport

2.2.2 Request / Response Layer Model

a) Piggy-backed: Client sends request using CON type or NON type message and receives response ACK with confirmable message immediately. In fig. 4, for successful response, ACK contain response message (identify by using token), for failure response, ACK contain failure response code[2].

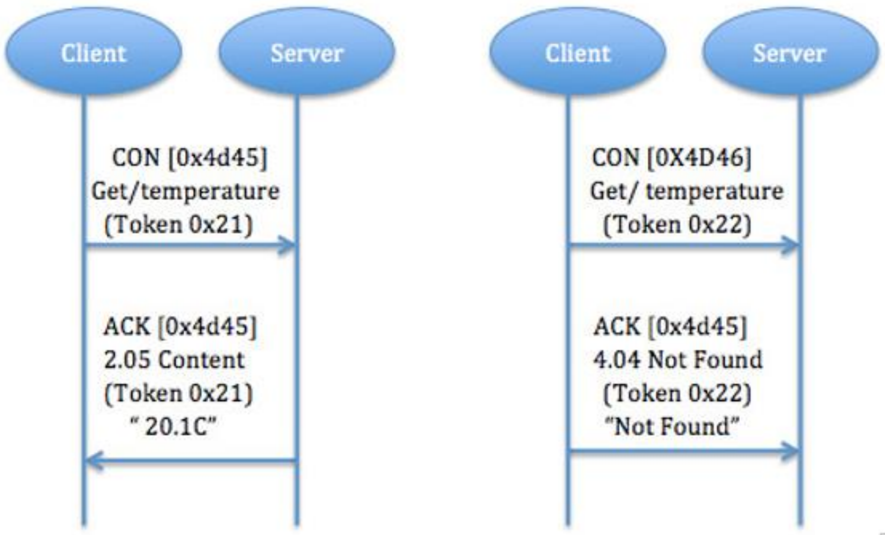


Fig. 5: The successful and failure response results of GET method

b) Separate response: If server receive a CON type message but not able to response this request immediately, it will send an empty ACK in case of client resend this message. When server ready to response this request, it will send a new CON to client and client reply a confirmable message with acknowledgment. ACK is just to confirm CON message, no matter CON message carry request or response (fig. 5).

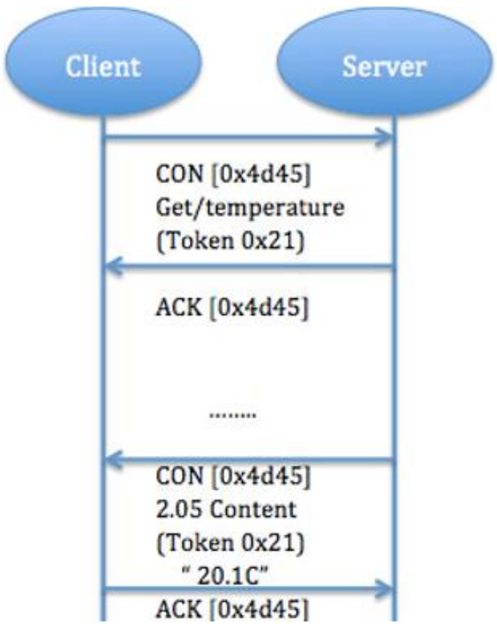


Fig:6 A Get request with a separate response

c) Non confirmable request and response: unlike Piggy-backed response carry confirmable message, in Non confirmable request client send NON type message indicate that Server don't need to confirm[2]. Server will resend a NON type message with response (fig. 6).

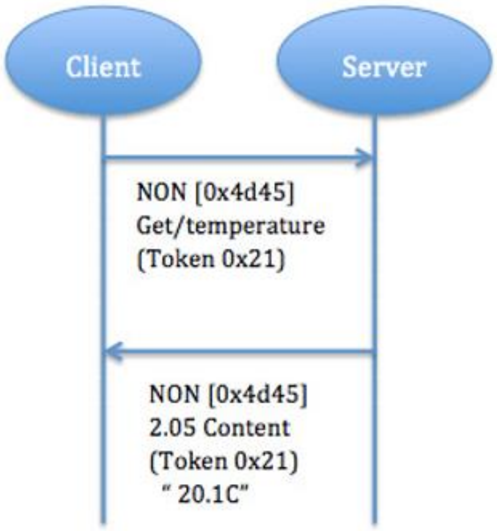


Fig:7 Non confirmable request and response

2.3 Message Format

CoAP is based on the exchange of compact messages that, by default, are transmitted over UDP (i.e. each CoAP message occupies the data section of one UDP datagram) [Petersburg12]. Message of CoAP uses simple binary format. Message= fixed-size 4-byte header plus a variable-length Token plus a sequence of CoAP options plus payload[2]. The format is shown in Table 2.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

Ver	T	OC	Code	MessageID
Token (if any, TKL bytes)…				
Options (if any)…				
Payload (if any)…				

Table 2:Message Format

3.HTTP /1.1 Protocol [RFC 2616] :-

The HTTP protocol is a request/response protocol. A client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over connection with a server. The server responds with a status line, including the message's protocol version and a success or error code followed by a MIME-like message containing server information, entity metainformation, and possible entity-body content[2].

HTTP is a communication protocol which is employed for delivering data (usually HTML files, multimedia files, etc.) on the World Wide Web through its default *TCP port 80*. However, there are other ports also which can be implemented for this function. HTTP has two different versions, HTTP/1.0, which is the old one and the newest HTTP/1.1. In its older version, a separate connection was required. In the case of a new version, the same connection can be recycled several times[2].

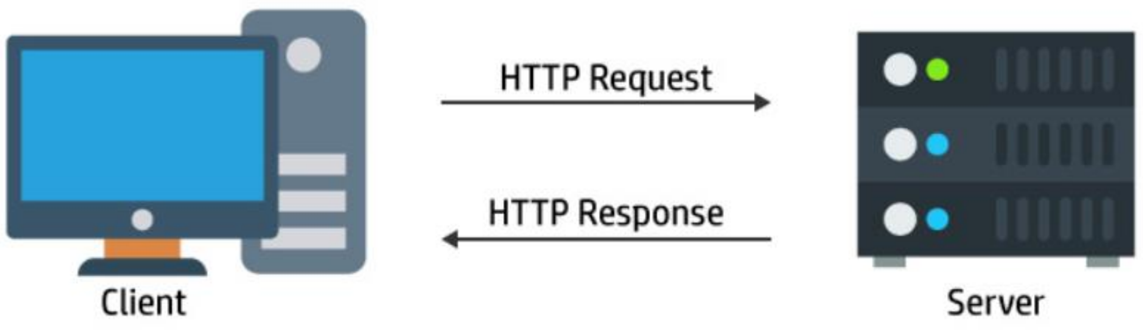


Fig:8 HTTP Protocol

The HTTP is meant for request/response depending on a client-server architecture where the user requests information through a web browser to the web server, which then responds to the requested data.

Web Client: The client of this client-server architecture asks for a request to a specific server through the HTTP (TCP/IP connection) as a request method in the form of a URL. It also contains a MIME-like message that contains request modifier and client information.

Web Server: This accepts the request and process with a response by a status line, together with the version of the message's protocol as well as the success or error code, followed by a MIME-like message having server information, some metadata, and possible the entity-body content holding the requested information.

3.1 Features of HTTP

- HTTP is connectionless: An HTTP request is initiated by the browser (HTTP client) as per the user's request for information. The server will process the request and launch back with a response which the client waits for.
- HTTP is simple: HTTP/2 does the encapsulation of HTTP messages into frames; i.e., HTTP is typically designed to be plain and human-readable.
- HTTP is extensible/customized: HTTP can be integrated with new functionality by providing a simple agreement between a client and a server.
- HTTP is stateless, but not session less: HTTP is stateless, which means there is no connection among two requests being consecutively carried out on the same connection. However, when the core of HTTP is itself a stateless one, HTTP cookies provide in making use of stateful sessions. Through the concept of header extensibility, HTTP cookies can be incorporated into the workflow, making session creation on each HTTP request for sharing the same content.

3.2 HTTP Methods

GET

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

HEAD

The HEAD method asks for a response identical to that of a GET request, but without the response body.

POST

The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.

PUT

The PUT method replaces all current representations of the target resource with the request payload.

DELETE

The DELETE method deletes the specified resource.

CONNECT

The CONNECT method establishes a tunnel to the server identified by the target resource.

OPTIONS

The OPTIONS method is used to describe the communication options for the target resource.

TRACE

The TRACE method performs a message loop-back test along the path to the target resource.

PATCH

The PATCH method is used to apply partial modifications to a resource.

4. CoAP Vs HTTP

- CoAP is network-oriented protocol, using similar features to HTTP but also allows for low overhead, multicast, etc. As HTTP protocol is a long-term successful standard, it can use small script to integrate various resources and services. Interoperation provided by HTTP is the key point of IoT, for this, HTTP is employed in application level. However, HTTP is based on TCP protocol using point to point (p2p) communication model that not suitable for notification push services. Also, for constrained devices, HTTP is too complex.
- Unlike HTTP based protocols, CoAP operates over UDP instead of using complex congestion control as in TCP CoAP is based on REST architecture, which is a general design for accessing Internet resources. In order to overcome disadvantage in constrained resource, CoAP need to optimize the length of datagram and provide reliable communication. On one side, CoAP provides URI, REST method such as GET, POST, PUT, and DELETE. On the other side, based on lightweight UDP protocol, CoAP allows IP multicast, which satisfies group communication for IoT. To compensate for the unreliability of UDP protocol, CoAP defines a retransmission mechanism and provides resource discovery mechanism with resource description Fig 1 shows the HTTP and CoAP protocol stacks[2].

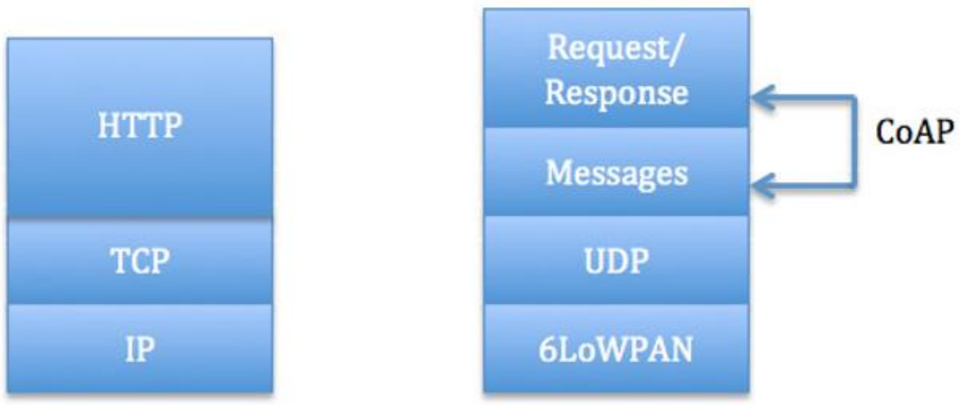


Fig: 9 HTTP and CoAP protocol stacks

CoAP is not just a simply compression of HTTP protocol. Considering low processing capability and low power consuming demand of restrained resource, CoAP redesigned some features of HTTP to accommodate these limitations. HTTP used under unconstrained network and CoAP used under constrained network.

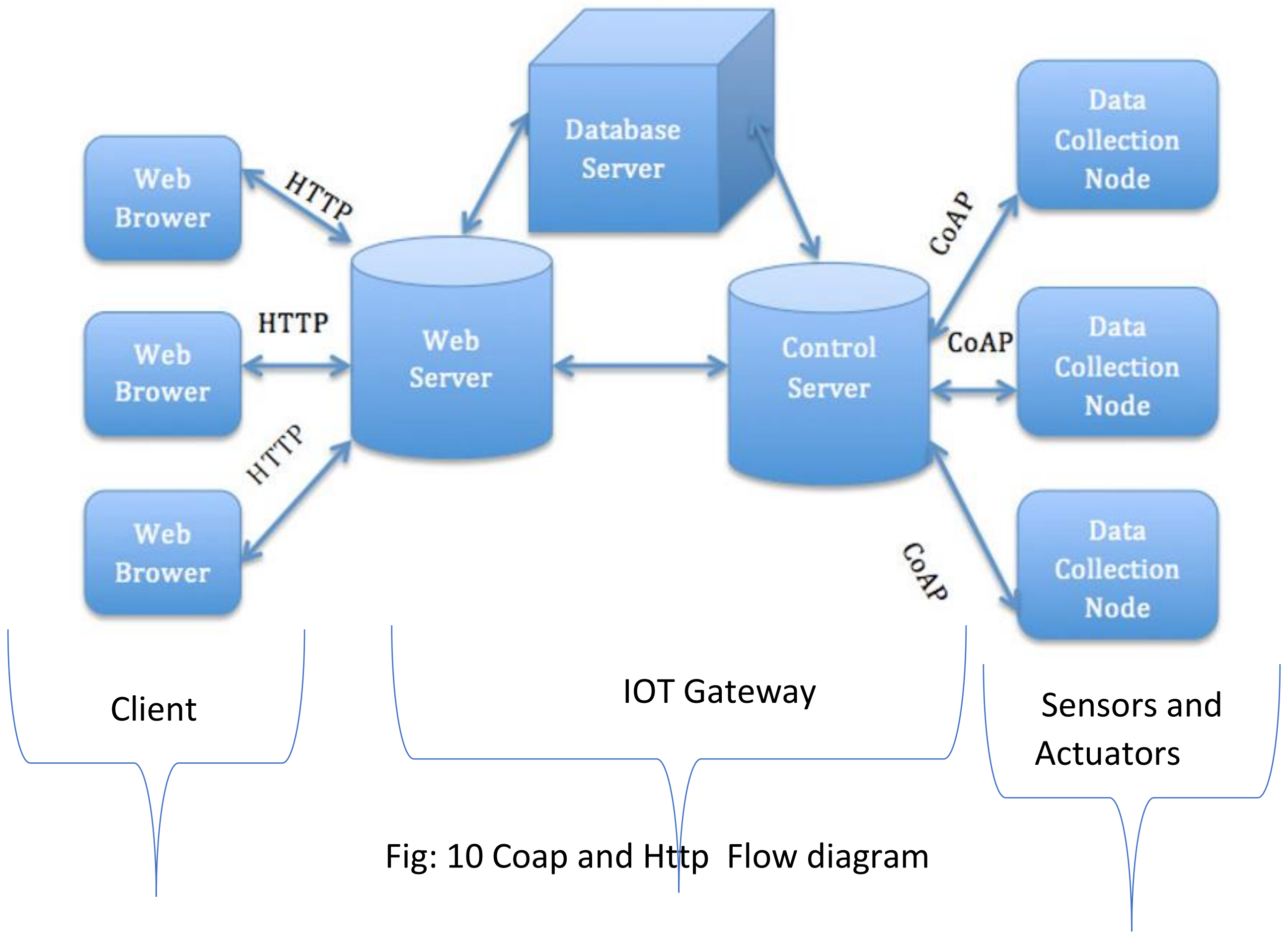


Fig: 10 Coap and Http Flow diagram

5 Project Implementation Description

IDE: Eclipse 2019-09 R (4.13.0)

Framework: Spring-Boot -version 2.2.2.Release

Backend Technologies: Core JAVA, J2EE, Inbuilt support for Tomcat webserver, XML, JSON

Frontend Technologies: HTML5, Bootstrap , JavaScript, Jquery

Protocol Implemented: HTTP/1.1 and CoAP(IOT)→Eclipse californium

Build Tool:- Maven 4.0.0 (Pom.xml)

5.1 Application Logic flow as per defined use-case:-

USE-CASE-1 Dog Safety:

CASE: A) Gate OPEN/CLOSE with respect to Dog Position Data Automatically

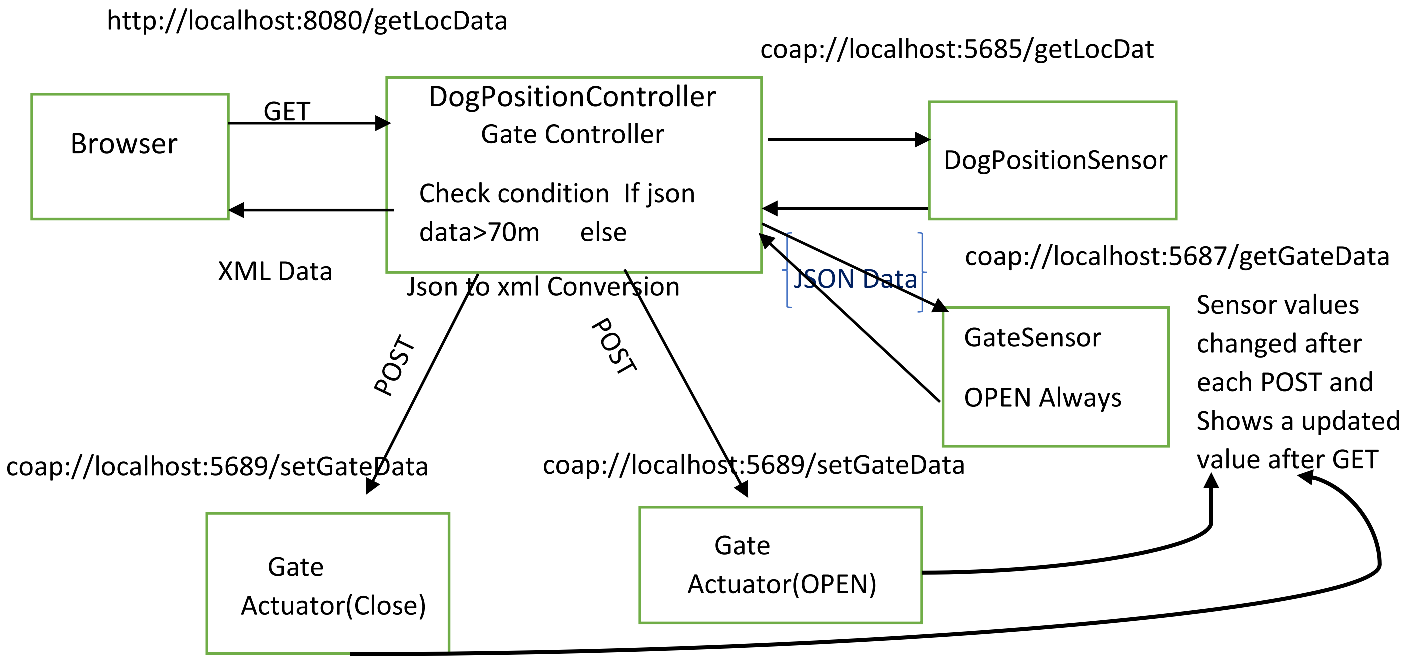

http://localhost:8080/getLocData
coap://localhost:5685/getLocDat
Browser
GET
XML Data
DogPositionController
Gate Controller
Check condition If json data>70m else
Json to xml Conversion
DogPositionSensor
JSON Data
coap://localhost:5687/getGateData
GateSensor
OPEN Always
Sensor values changed after each POST and Shows a updated value after GET
POST
POST
coap://localhost:5689/setGateData
coap://localhost:5689/setGateData
Gate Actuator(Close)
Gate Actuator(OPEN)

Fig: 11 DATA FLOW BLOCK DIAGRAM

Explanation:- Browser will get a continuous value of dog position data from dog position sensor file (coap://localhost:5685/getLocData) and continuous value of Gate data from Gate sensor file (coap://localhost:5687/getGateData) which is by default OPEN but if the dog data go beyond 70 m (as per my defined use case) then Gate will trigger POST to close the gate (coap://localhost:5689/setGateData) and the Gate sensor file will update CLOSE in turn display Close status on Browser.

CASE: B) Alarm ON/OFF with respect to Dog Position Data Automatically

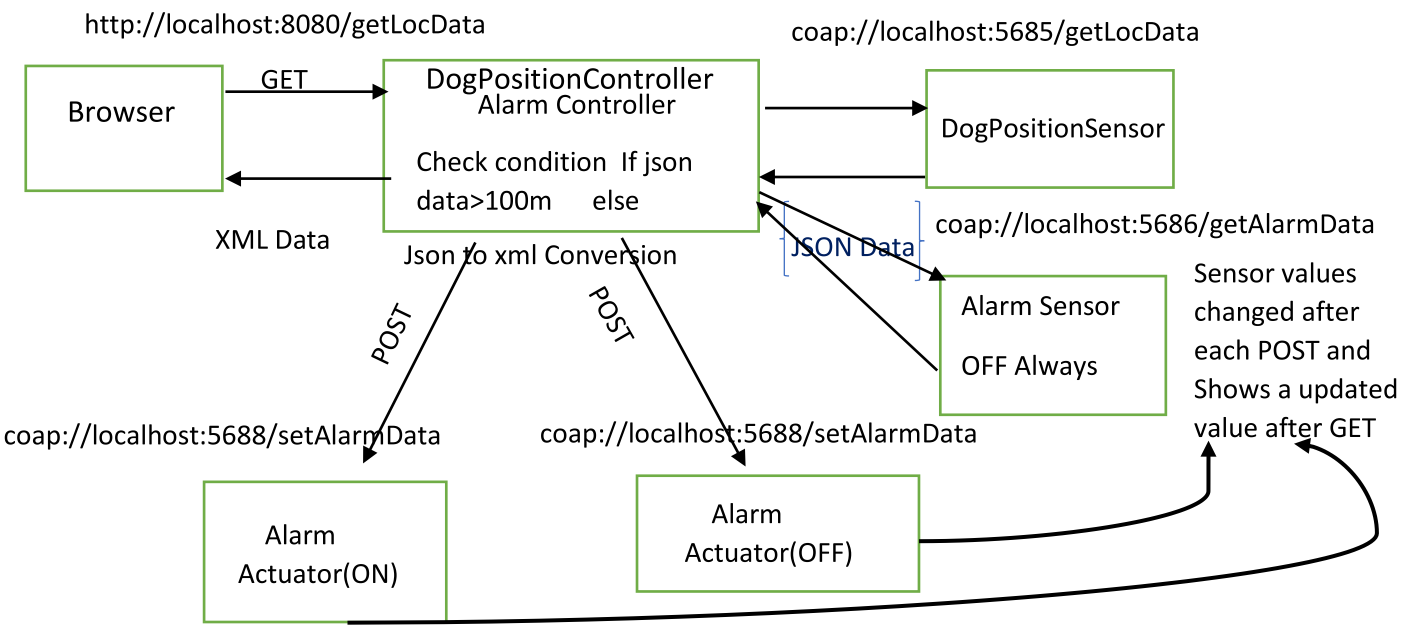

http://localhost:8080/getLocData
coap://localhost:5685/getLocData
Browser
GET
DogPositionController
Alarm Controller
Check condition If json data>100m else
DogPositionSensor
XML Data
Json to xml Conversion
JSON Data
coap://localhost:5686/getAlarmData
Alarm Sensor
OFF Always
Sensor values changed after each POST and Shows a updated value after GET
POST
POST
coap://localhost:5688/setAlarmData
coap://localhost:5688/setAlarmData
Alarm Actuator(ON)
Alarm Actuator(OFF)

Fig: 12 DATA FLOW BLOCK DIAGRAM

Explanation:- Browser will get a continuous value of dog position data from dog position sensor file (coap://localhost:5685/getLocData) and continuous value of Alarm data from Alarm sensor file (coap://localhost:5686/getAlarmData) which is by default OFF but if the dog data go beyond 100 m (as per my defined use case) then Alarm will trigger POST to ON the Alarm (coap://localhost:5688/setGateData) and the Alarm sensor file will update ON in turn display ON status on Browser.

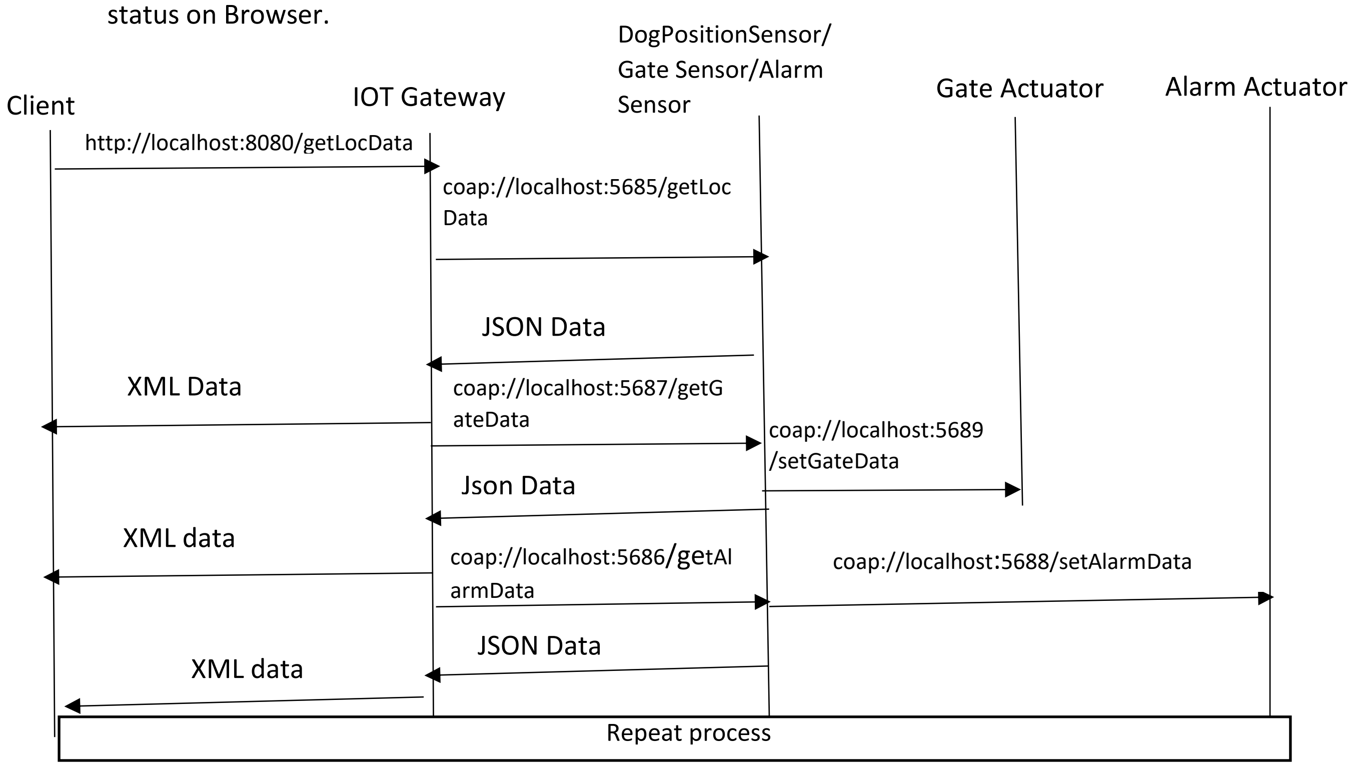
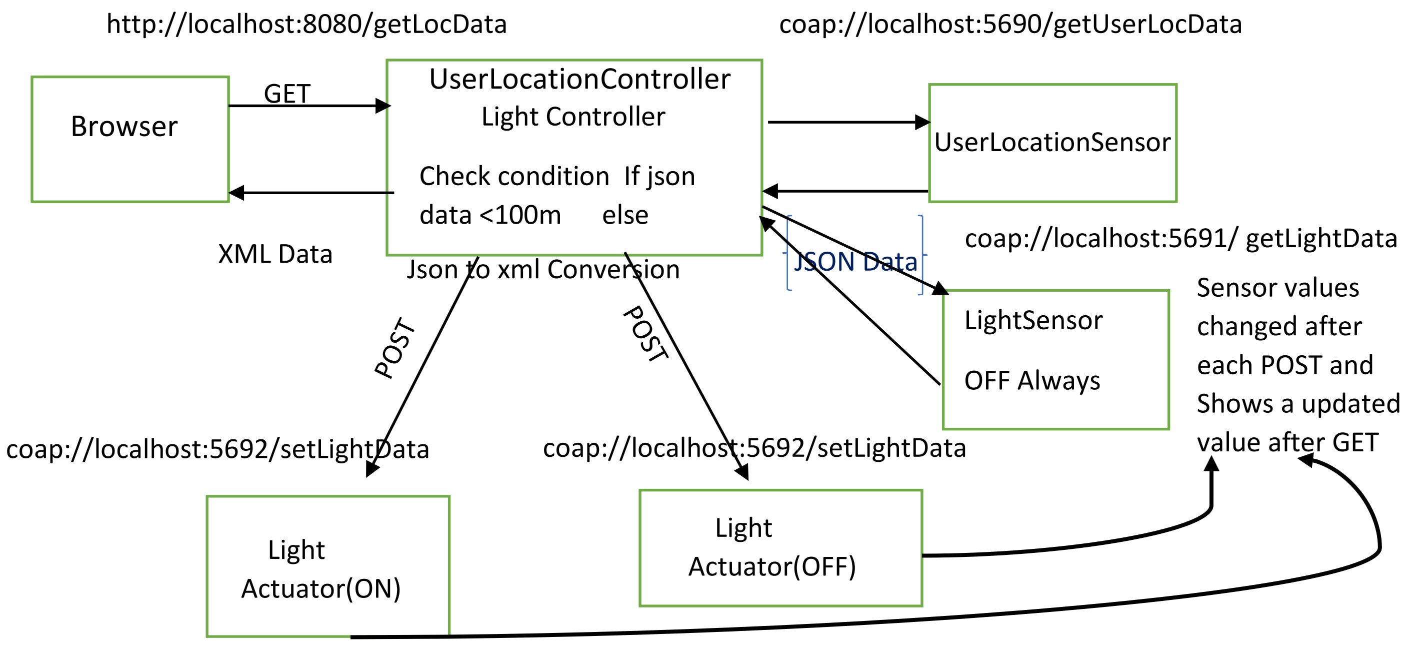

Client
IOT Gateway
DogPositionSensor/ Gate Sensor/Alarm Sensor
Gate Actuator
Alarm Actuator
http://localhost:8080/getLocData
coap://localhost:5685/getLoc Data
JSON Data
XML Data
coap://localhost:5687/getG ateData
coap://localhost:5689 /setGateData
Json Data
XML data
coap://localhost:5686/getAl armData
coap://localhost:5688/setAlarmData
JSON Data
XML data
Repeat process

FIG:13 MSG Sequence Diagram of Case A and Case B discussed above

USE-CASE-2 Power saving via Lights and ventilation via Window:-

CASE: A) Lights ON/OFF with respect to User Location Data Automatically


http://localhost:8080/getLocData
coap://localhost:5690/getUserLocData
Browser
GET
UserLocationController
Light Controller
Check condition If json data <100m else
Json to xml Conversion
XML Data
UserLocationSensor
coap://localhost:5691/ getLightData
JSON Data
LightSensor
OFF Always
Sensor values changed after each POST and Shows a updated value after GET
POST
POST
coap://localhost:5692/setLightData
coap://localhost:5692/setLightData
Light Actuator(ON)
Light Actuator(OFF)

Explanation:- Browser will get a continuous value of User Location data from User Location sensor file (coap://localhost:5690/getUserLocData) and continuous value of Light data from Light sensor file (coap://localhost:5691/getLightData) which is by default OFF but if the User data shows nearby home (<100 m as per my defined use case) then Lights will trigger POST to ON the Bulb(Light)(coap://localhost:5692/setLightData) and the Light sensor file will update ON in turn display ON status on Browser.

CASE:B) Window OPEN/CLOSE with respect to UserLocationData Automatically

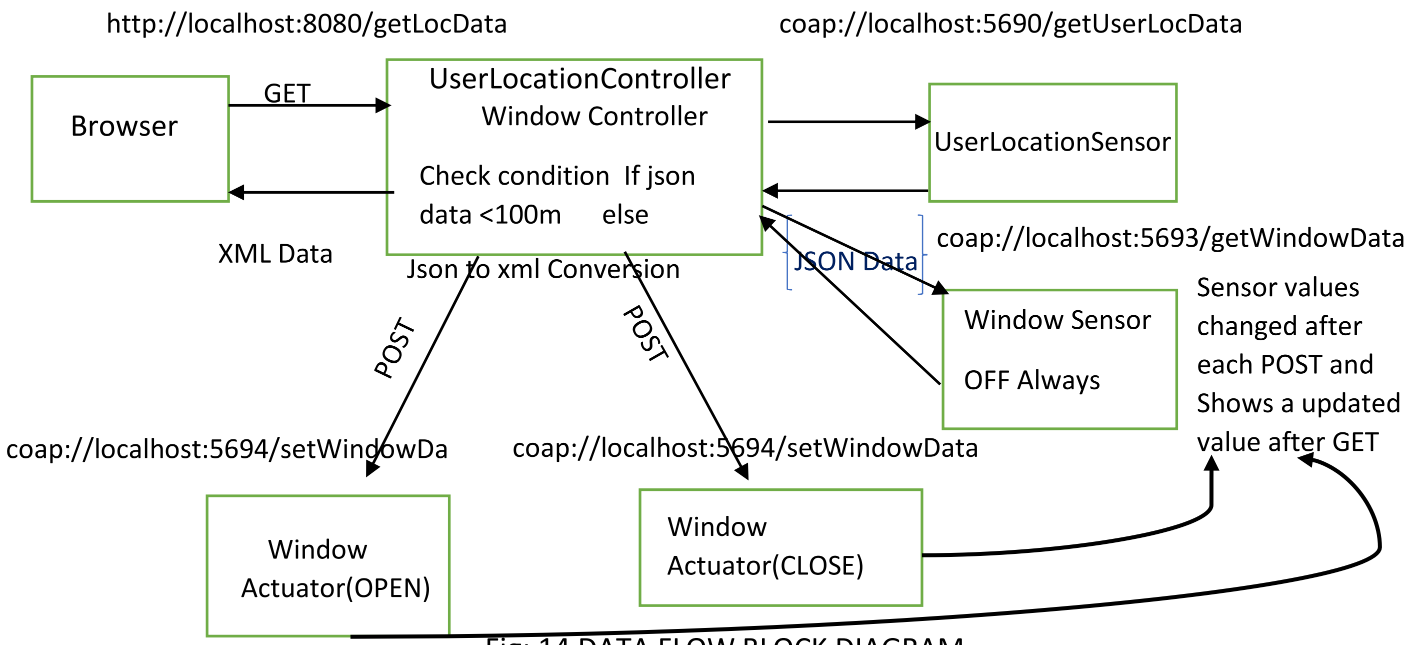

http://localhost:8080/getLocData
coap://localhost:5690/getUserLocData
Browser
GET
UserLocationController
Window Controller
Check condition If json data <100m else
Json to xml Conversion
XML Data
UserLocationSensor
coap://localhost:5693/getWindowData
JSON Data
Window Sensor
OFF Always
Sensor values changed after each POST and Shows a updated value after GET
POST
POST
coap://localhost:5694/setWindowDa
coap://localhost:5694/setWindowData
Window Actuator(OPEN)
Window Actuator(CLOSE)

Fig: 14 DATA FLOW BLOCK DIAGRAM

Explanation:- Browser will get a continuous value of User Location data from User Location sensor file (coap://localhost:5690/getUserLocData) and continuous value of Window data from Window sensor file (coap://localhost:5693/getWindowData) which is by default OFF but if the User data shows nearby home (<100 m as per my defined use case) then Window will trigger POST to OPEN the Window (coap://localhost:5694/setWindowData) and the Window sensor file will update OPEN in turn display ON status on Browser.

USE-CASE-3 Kids Safety:-

CASE:- Set Kids Location Manually to control kids range from Home and Trigger an alert in case Kids move out of defined Location Automatically

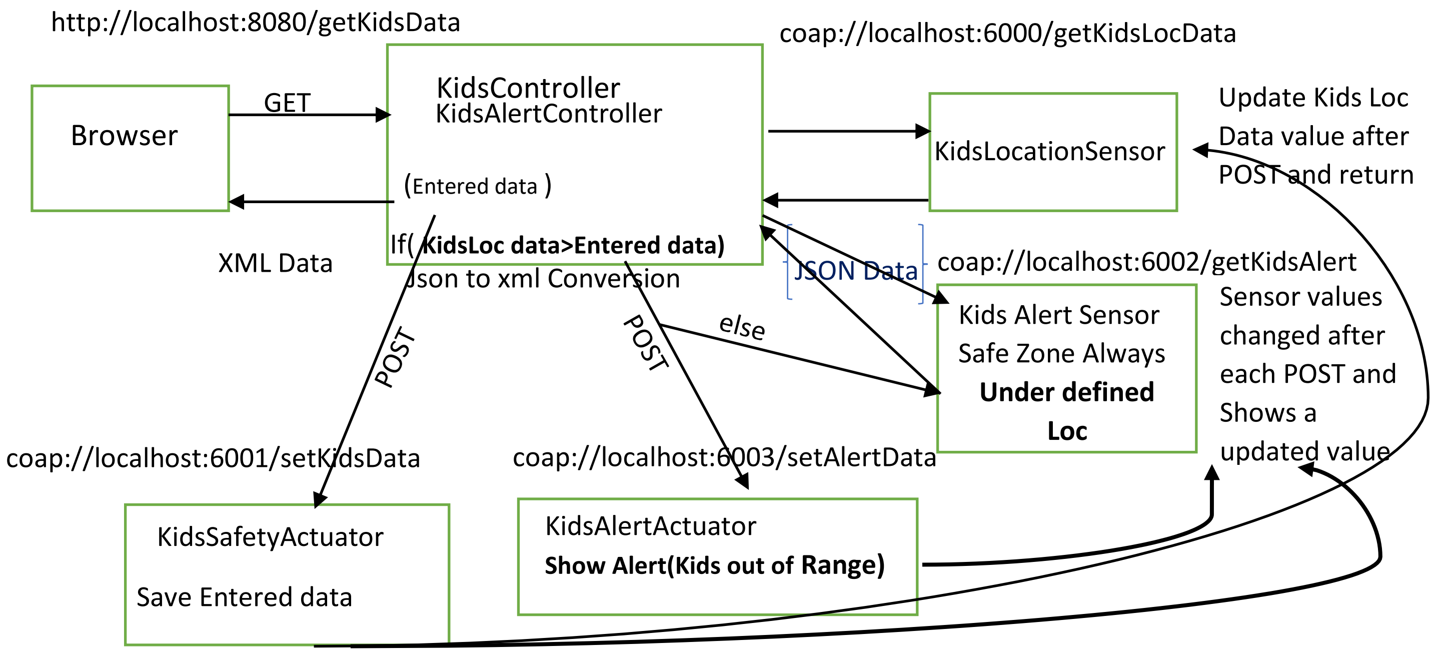
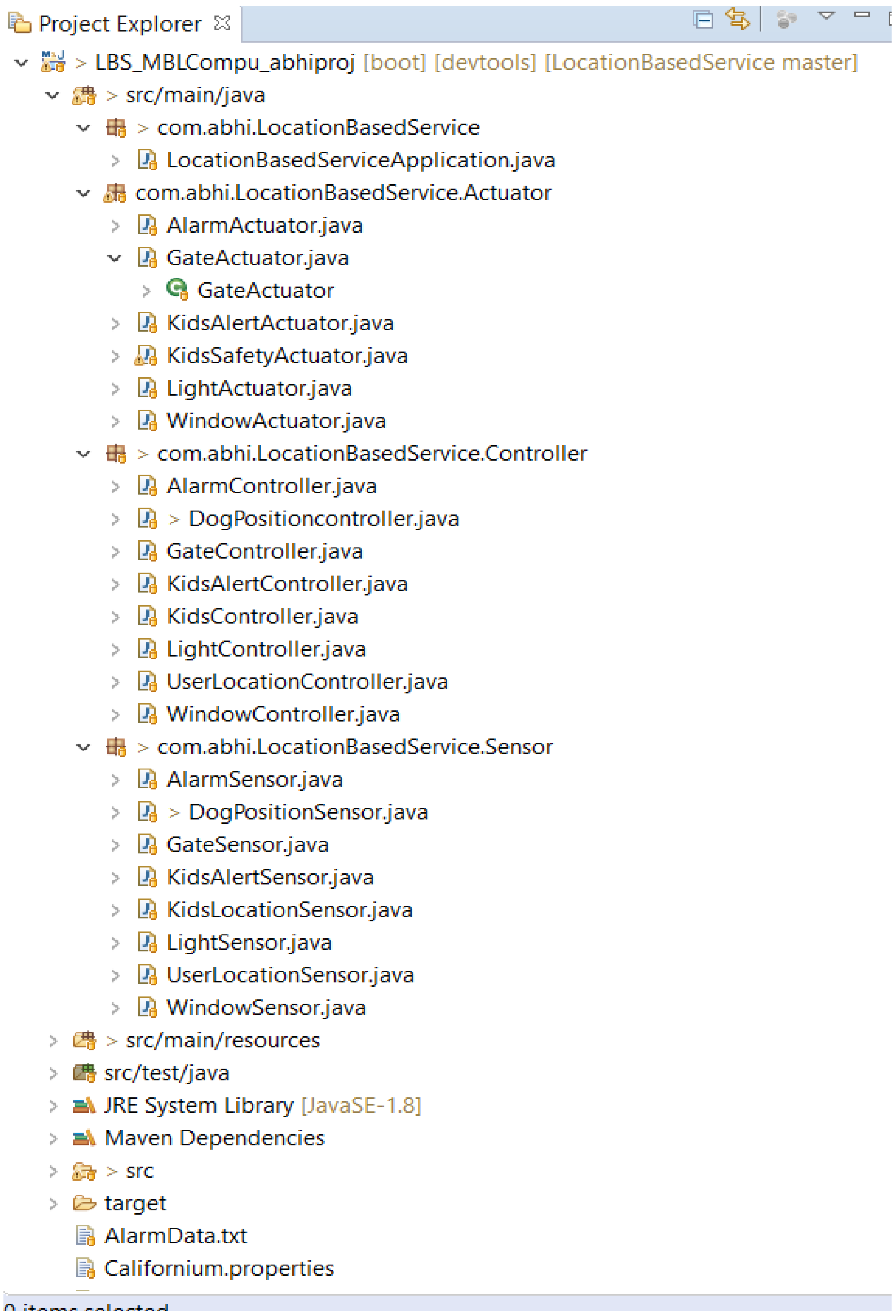

http://localhost:8080/getKidsData
coap://localhost:6000/getKidsLocData
Browser
GET
KidsController
KidsAlertController
(Entered data)
If(KidsLoc data>Entered data)
Json to xml Conversion
XML Data
KidsLocationSensor
Update Kids Loc Data value after POST and return
JSON Data
coap://localhost:6002/getKidsAlert
Kids Alert Sensor
Safe Zone Always
Under defined Loc
Sensor values changed after each POST and Shows a updated value
else
POST
POST
coap://localhost:6001/setKidsData
coap://localhost:6003/setAlertData
KidsSafetyActuator
Save Entered data
KidsAlertActuator
Show Alert(Kids out of Range)

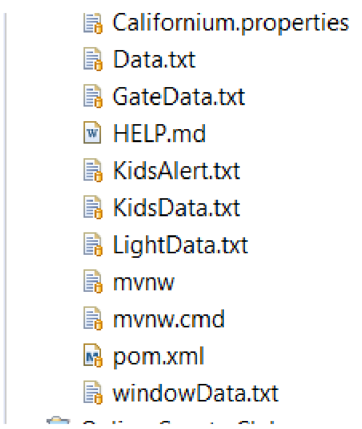
Fig: 15 DATA FLOW BLOCK DIAGRAM

Explanation:- In this case, First I need to set Kids Location Manually(compulsory step as per my use case) on button click, then alert will show in case of kids location data go beyond Entered data else kids under safe zone(defined location) will be shown.

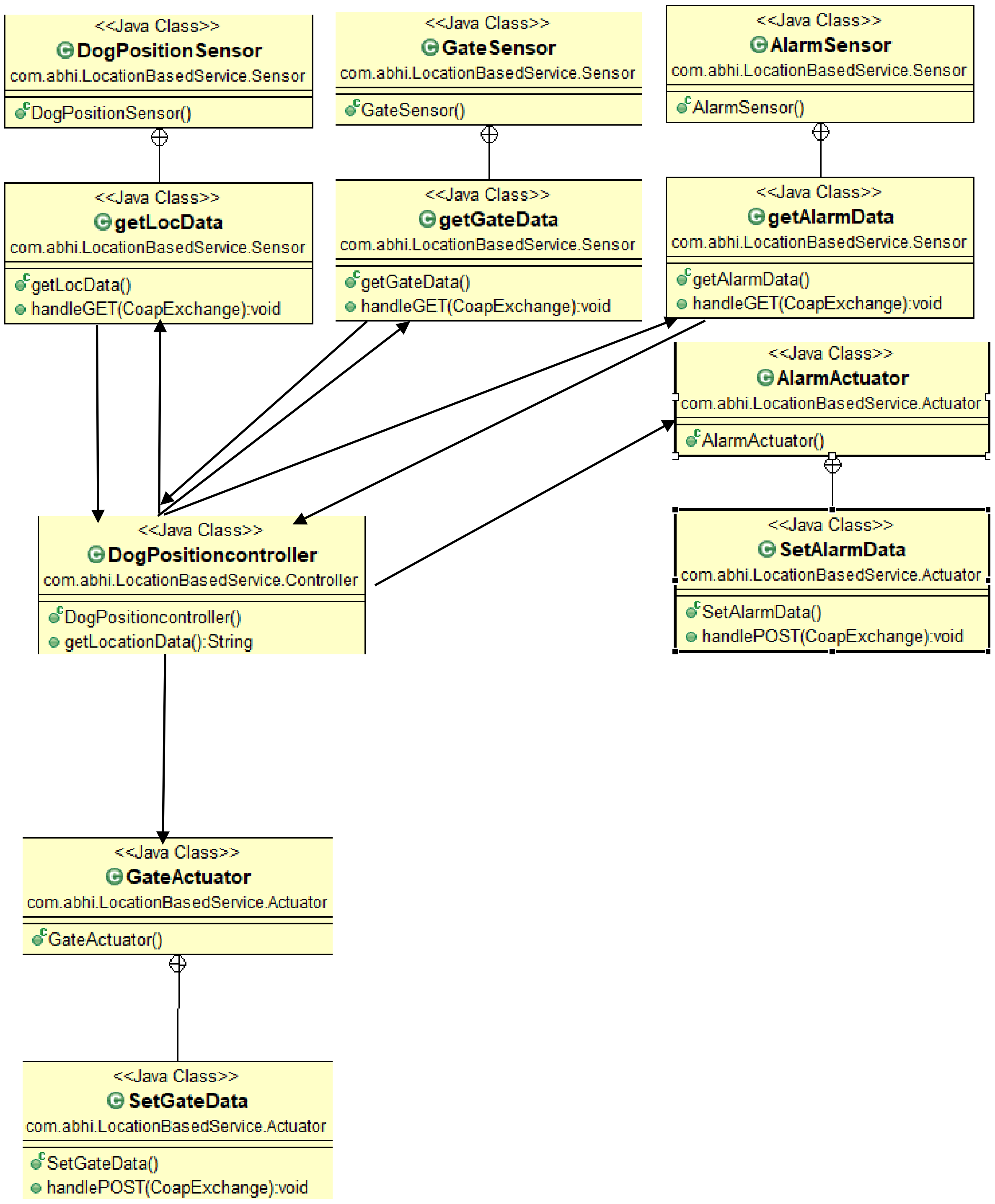
6.Project Structure and UML Class diagram



```
Project Explorer
v > LBS_MBLCompu_abhiproj [boot] [devtools] [LocationBasedService master]
  v > src/main/java
    v > com.abhi.LocationBasedService
      > LocationBasedServiceApplication.java
    v com.abhi.LocationBasedService.Actuator
      > AlarmActuator.java
      v GateActuator.java
        > GateActuator
      > KidsAlertActuator.java
      > KidsSafetyActuator.java
      > LightActuator.java
      > WindowActuator.java
    v > com.abhi.LocationBasedService.Controller
      > AlarmController.java
      > > DogPositioncontroller.java
      > GateController.java
      > KidsAlertController.java
      > KidsController.java
      > LightController.java
      > UserLocationController.java
      > WindowController.java
    v > com.abhi.LocationBasedService.Sensor
      > AlarmSensor.java
      > > DogPositionSensor.java
      > GateSensor.java
      > KidsAlertSensor.java
      > KidsLocationSensor.java
      > LightSensor.java
      > UserLocationSensor.java
      > WindowSensor.java
  > > src/main/resources
  > src/test/java
  > JRE System Library [JavaSE-1.8]
  > Maven Dependencies
  > > src
  > target
    AlarmData.txt
    Californium.properties
0 items selected
```



```
    Californium.properties
    Data.txt
    GateData.txt
    HELP.md
    KidsAlert.txt
    KidsData.txt
    LightData.txt
    mvnw
    mvnw.cmd
    pom.xml
    windowData.txt
```



<<Java Class>>
DogPositionSensor
com.abhi.LocationBasedService.Sensor
DogPositionSensor()
<<Java Class>>
GateSensor
com.abhi.LocationBasedService.Sensor
GateSensor()
<<Java Class>>
AlarmSensor
com.abhi.LocationBasedService.Sensor
AlarmSensor()
<<Java Class>>
getLocData
com.abhi.LocationBasedService.Sensor
getLocData()
handleGET(CoapExchange):void
<<Java Class>>
getGateData
com.abhi.LocationBasedService.Sensor
getGateData()
handleGET(CoapExchange):void
<<Java Class>>
getAlarmData
com.abhi.LocationBasedService.Sensor
getAlarmData()
handleGET(CoapExchange):void
<<Java Class>>
AlarmActuator
com.abhi.LocationBasedService.Actuator
AlarmActuator()
<<Java Class>>
SetAlarmData
com.abhi.LocationBasedService.Actuator
SetAlarmData()
handlePOST(CoapExchange):void
<<Java Class>>
DogPositioncontroller
com.abhi.LocationBasedService.Controller
DogPositioncontroller()
getLocationData():String
<<Java Class>>
GateActuator
com.abhi.LocationBasedService.Actuator
GateActuator()
<<Java Class>>
SetGateData
com.abhi.LocationBasedService.Actuator
SetGateData()
handlePOST(CoapExchange):void

Fig: UML Class Diagram(Use case-1)

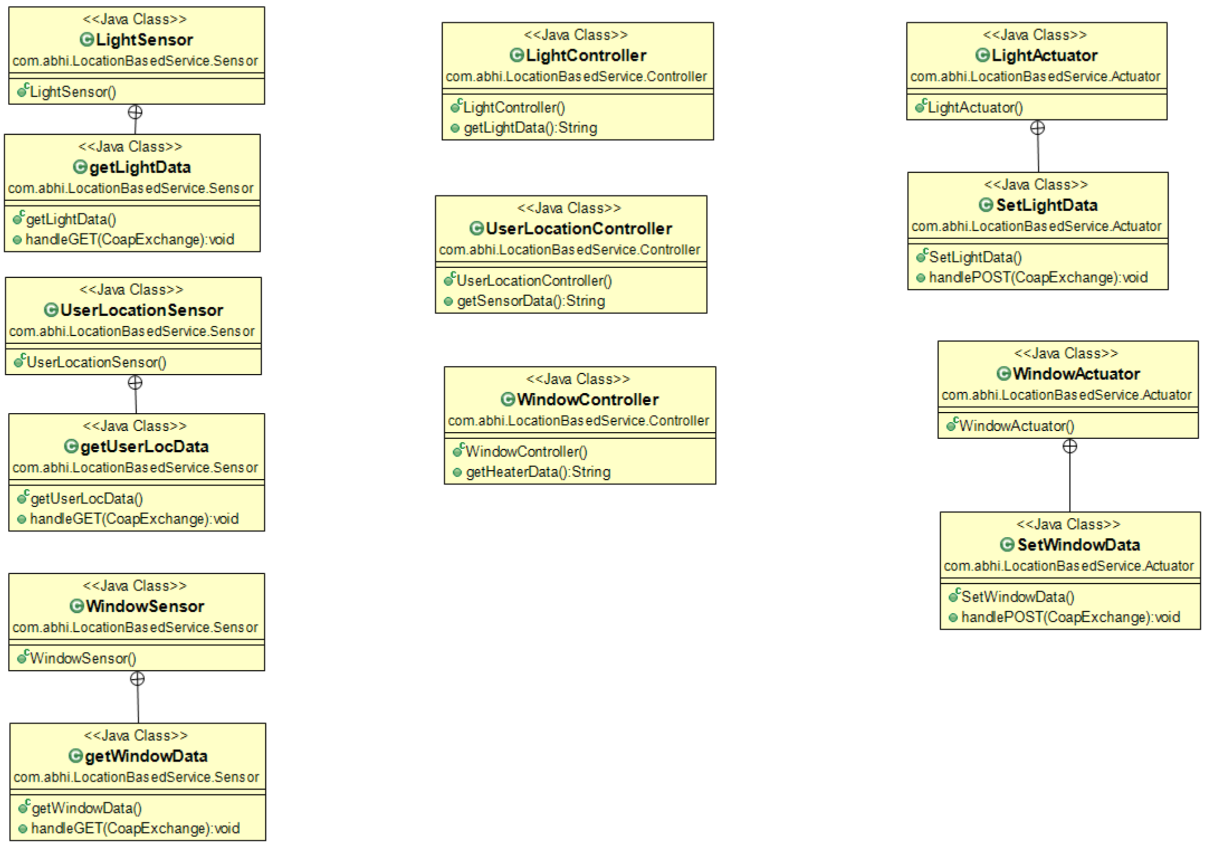


Fig: UML class diagram(use case:2) showing only classes components

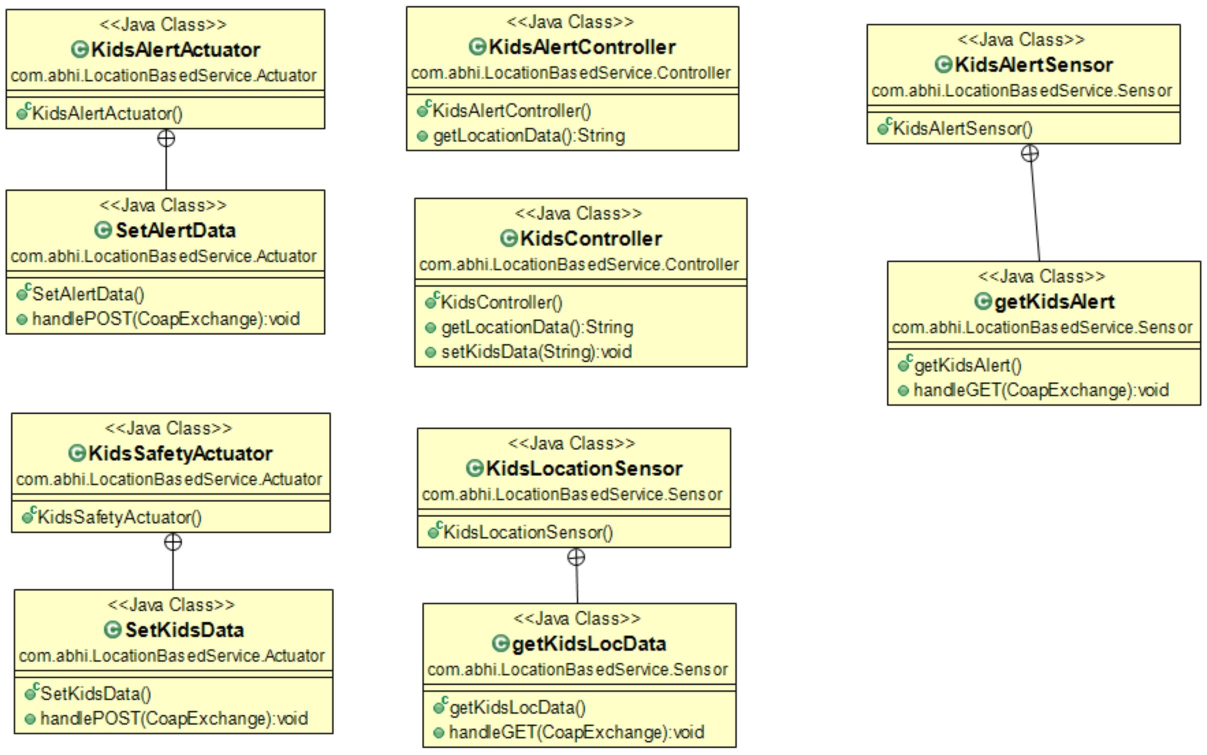
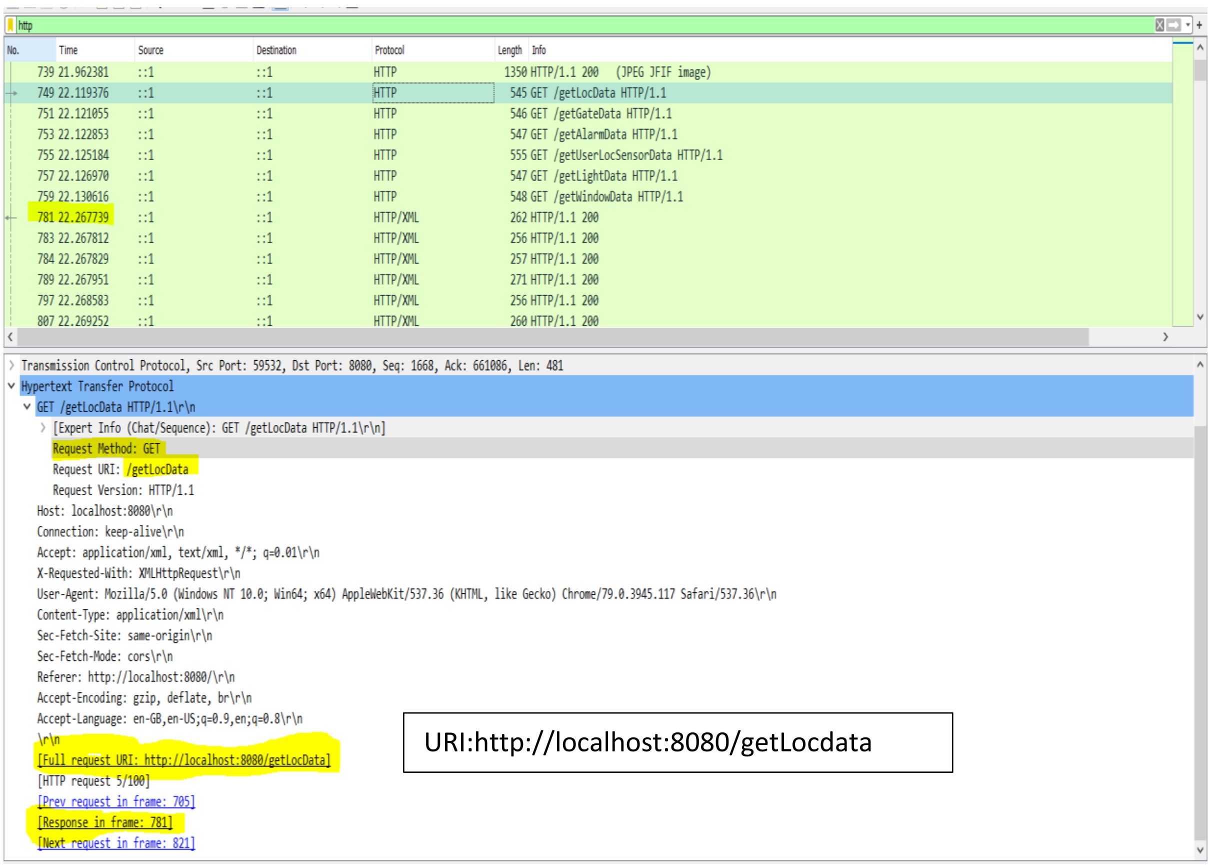


Fig:. UML Class Diagram (Use-case:3) showing only class components

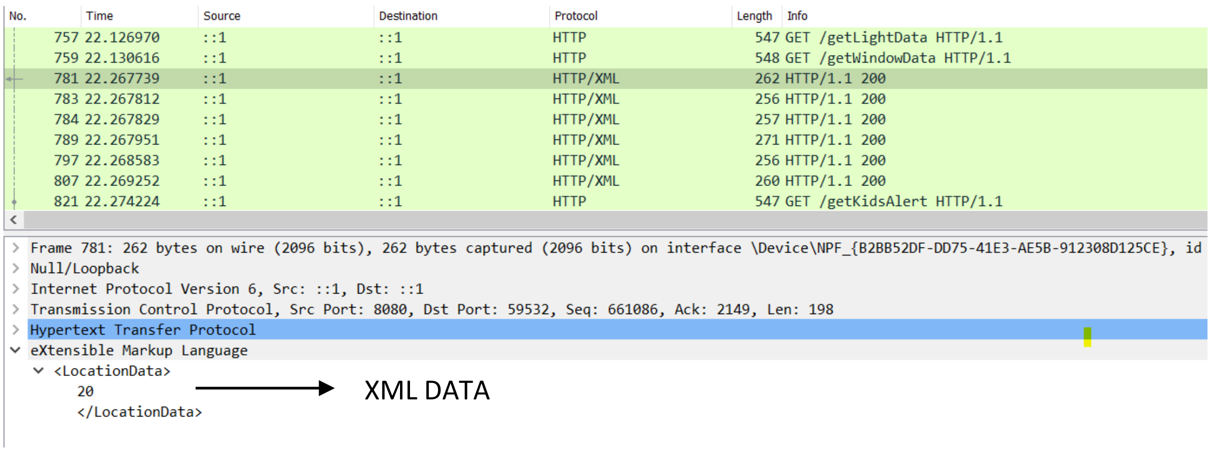
7. WireShark Capture:-



```
http

No.   Time          Source     Destination   Protocol   Length Info
  739 21.962381     ::1        ::1           HTTP       1350 HTTP/1.1 200   (JPEG JFIF image)
  749 22.119376     ::1        ::1           HTTP        545 GET /getLocData HTTP/1.1
  751 22.121055     ::1        ::1           HTTP        546 GET /getGateData HTTP/1.1
  753 22.122853     ::1        ::1           HTTP        547 GET /getAlarmData HTTP/1.1
  755 22.125184     ::1        ::1           HTTP        555 GET /getUserLocSensorData HTTP/1.1
  757 22.126970     ::1        ::1           HTTP        547 GET /getLightData HTTP/1.1
  759 22.130616     ::1        ::1           HTTP        548 GET /getWindowData HTTP/1.1
  781 22.267739     ::1        ::1           HTTP/XML    262 HTTP/1.1 200
  783 22.267812     ::1        ::1           HTTP/XML    256 HTTP/1.1 200
  784 22.267829     ::1        ::1           HTTP/XML    257 HTTP/1.1 200
  789 22.267951     ::1        ::1           HTTP/XML    271 HTTP/1.1 200
  797 22.268583     ::1        ::1           HTTP/XML    256 HTTP/1.1 200
  807 22.269252     ::1        ::1           HTTP/XML    260 HTTP/1.1 200

> Transmission Control Protocol, Src Port: 59532, Dst Port: 8080, Seq: 1668, Ack: 661086, Len: 481
v Hypertext Transfer Protocol
  v GET /getLocData HTTP/1.1\r\n
    > [Expert Info (Chat/Sequence): GET /getLocData HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /getLocData
      Request Version: HTTP/1.1
    Host: localhost:8080\r\n
    Connection: keep-alive\r\n
    Accept: application/xml, text/xml, */*; q=0.01\r\n
    X-Requested-With: XMLHttpRequest\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36\r\n
    Content-Type: application/xml\r\n
    Sec-Fetch-Site: same-origin\r\n
    Sec-Fetch-Mode: cors\r\n
    Referer: http://localhost:8080/\r\n
    Accept-Encoding: gzip, deflate, br\r\n
    Accept-Language: en-GB,en-US;q=0.9,en;q=0.8\r\n
    \r\n
    [Full request URI: http://localhost:8080/getLocData]
    [HTTP request 5/100]
    [Prev request in frame: 705]
    [Response in frame: 781]
    [Next request in frame: 821]
```

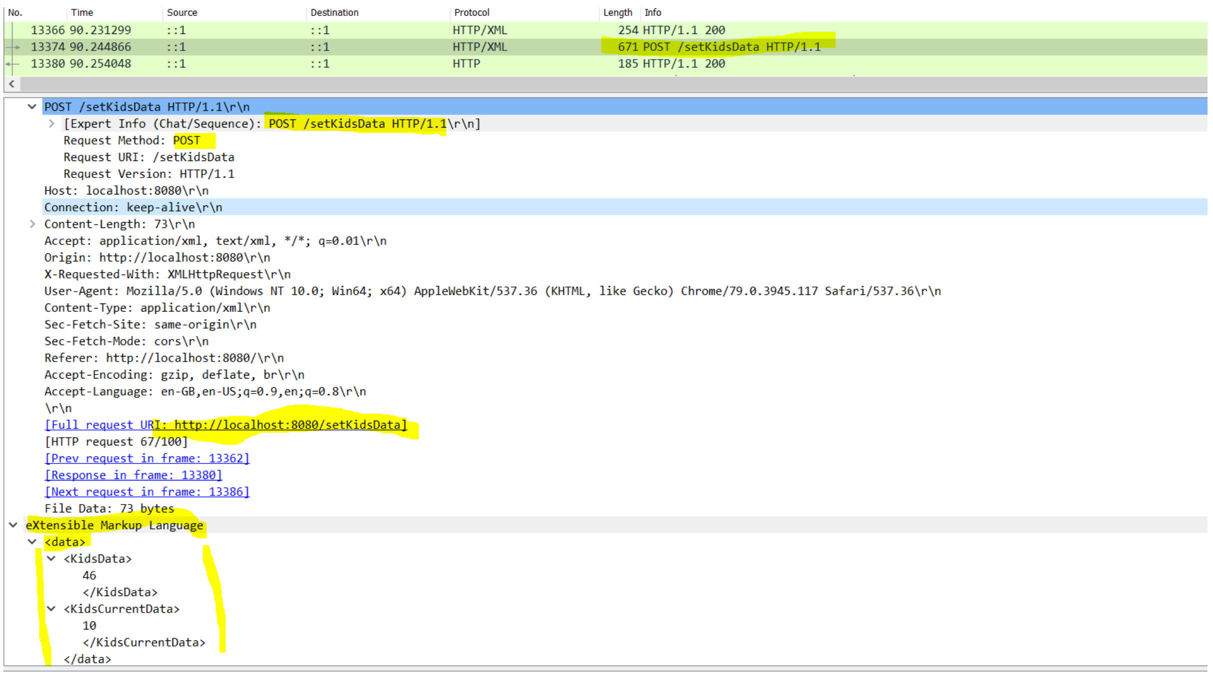
URI:http://localhost:8080/getLocdata

Fig: showing HTTP Get Request

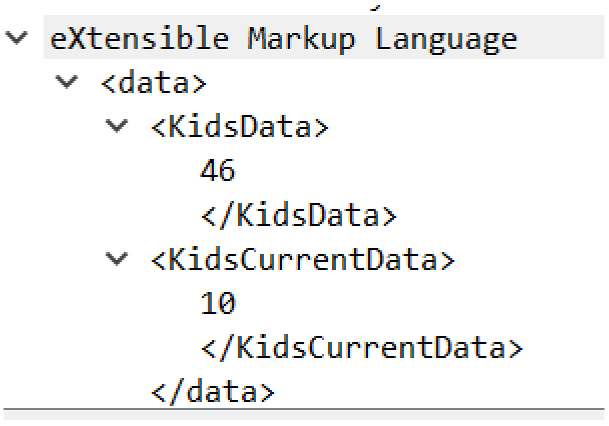


```
No.   Time          Source     Destination   Protocol   Length Info
  757 22.126970     ::1        ::1           HTTP        547 GET /getLightData HTTP/1.1
  759 22.130616     ::1        ::1           HTTP        548 GET /getWindowData HTTP/1.1
  781 22.267739     ::1        ::1           HTTP/XML    262 HTTP/1.1 200
  783 22.267812     ::1        ::1           HTTP/XML    256 HTTP/1.1 200
  784 22.267829     ::1        ::1           HTTP/XML    257 HTTP/1.1 200
  789 22.267951     ::1        ::1           HTTP/XML    271 HTTP/1.1 200
  797 22.268583     ::1        ::1           HTTP/XML    256 HTTP/1.1 200
  807 22.269252     ::1        ::1           HTTP/XML    260 HTTP/1.1 200
  821 22.274224     ::1        ::1           HTTP        547 GET /getKidsAlert HTTP/1.1

> Frame 781: 262 bytes on wire (2096 bits), 262 bytes captured (2096 bits) on interface \Device\NPF_{B2BB52DF-DD75-41E3-AE5B-912308D125CE}, id
> Null/Loopback
> Internet Protocol Version 6, Src: ::1, Dst: ::1
> Transmission Control Protocol, Src Port: 8080, Dst Port: 59532, Seq: 661086, Ack: 2149, Len: 198
> Hypertext Transfer Protocol
v eXtensible Markup Language
  v <LocationData>
      20
    </LocationData>
```

Fig: showing HTTP Response

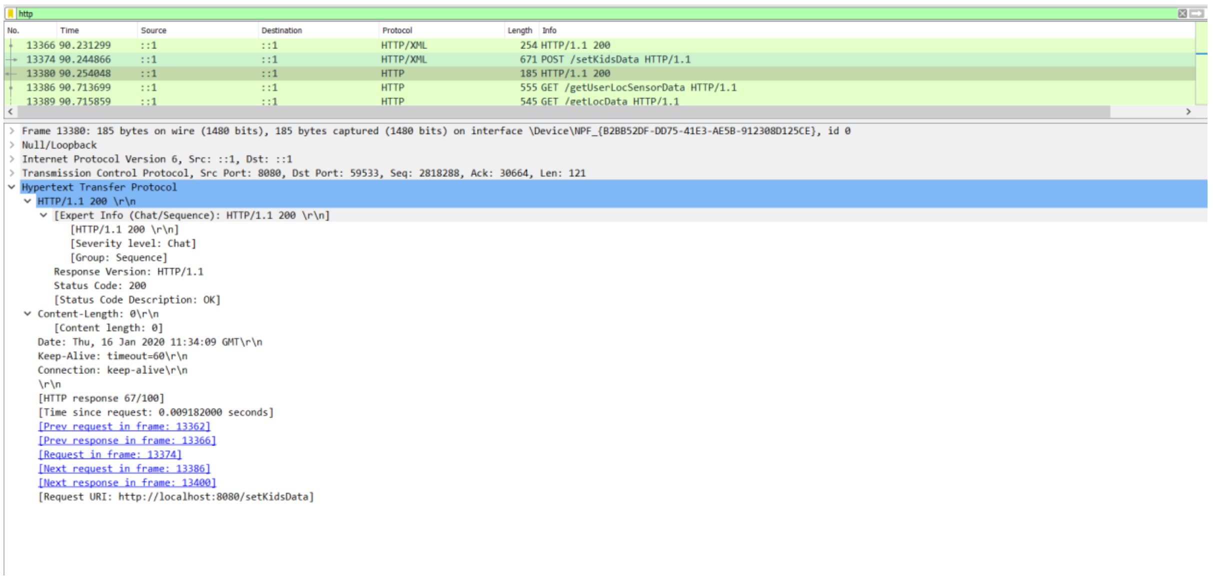


```
No.   Time        Source      Destination   Protocol    Length  Info
13366 90.231299   ::1         ::1           HTTP/XML       254 HTTP/1.1 200
13374 90.244866   ::1         ::1           HTTP/XML       671 POST /setKidsData HTTP/1.1
13380 90.254048   ::1         ::1           HTTP           185 HTTP/1.1 200

v POST /setKidsData HTTP/1.1\r\n
  > [Expert Info (Chat/Sequence): POST /setKidsData HTTP/1.1\r\n]
    Request Method: POST
    Request URI: /setKidsData
    Request Version: HTTP/1.1
  Host: localhost:8080\r\n
  Connection: keep-alive\r\n
> Content-Length: 73\r\n
  Accept: application/xml, text/xml, */*; q=0.01\r\n
  Origin: http://localhost:8080\r\n
  X-Requested-With: XMLHttpRequest\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36\r\n
  Content-Type: application/xml\r\n
  Sec-Fetch-Site: same-origin\r\n
  Sec-Fetch-Mode: cors\r\n
  Referer: http://localhost:8080/\r\n
  Accept-Encoding: gzip, deflate, br\r\n
  Accept-Language: en-GB,en-US;q=0.9,en;q=0.8\r\n
  \r\n
  [Full request URI: http://localhost:8080/setKidsData]
  [HTTP request 67/100]
  [Prev request in frame: 13362]
  [Response in frame: 13380]
  [Next request in frame: 13386]
  File Data: 73 bytes
v eXtensible Markup Language
  v <data>
    v <KidsData>
        46
        </KidsData>
    v <KidsCurrentData>
        10
        </KidsCurrentData>
      </data>
```



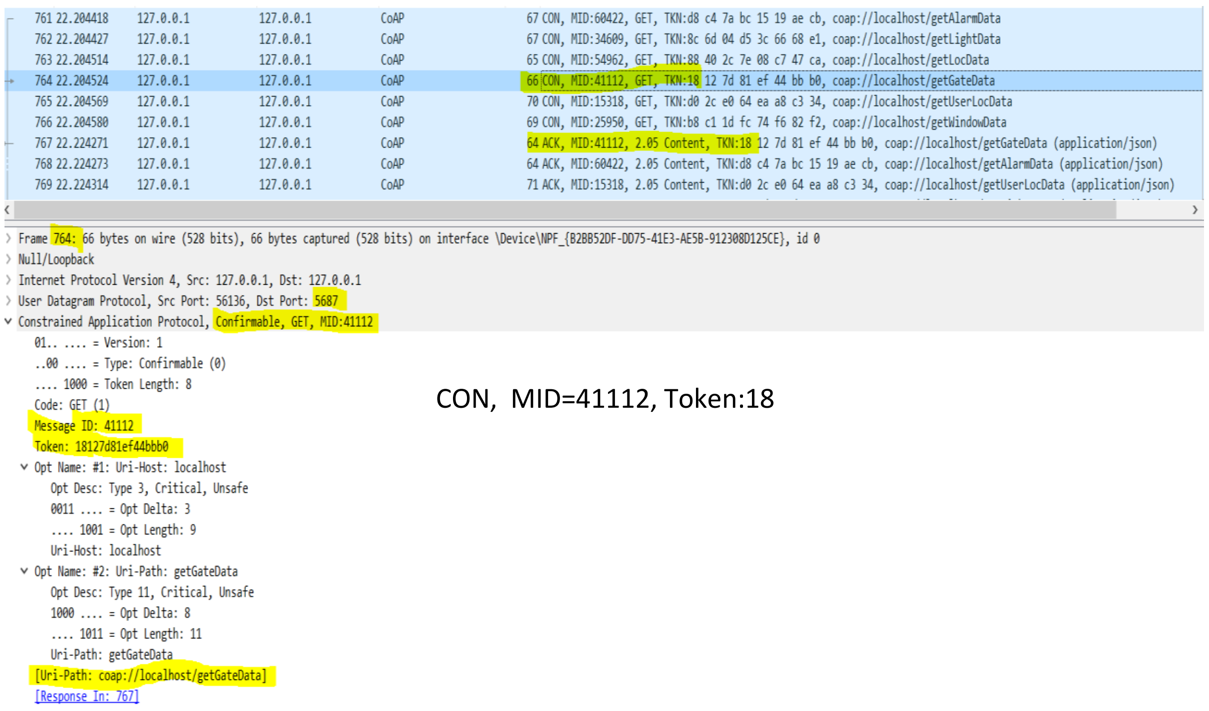
```
v eXtensible Markup Language
  v <data>
    v <KidsData>
        46
        </KidsData>
    v <KidsCurrentData>
        10
        </KidsCurrentData>
      </data>
```

Fig: HTTP POST Request



```
http
No.   Time        Source      Destination   Protocol    Length  Info
13366 90.231299   ::1         ::1           HTTP/XML       254 HTTP/1.1 200
13374 90.244866   ::1         ::1           HTTP/XML       671 POST /setKidsData HTTP/1.1
13380 90.254048   ::1         ::1           HTTP           185 HTTP/1.1 200
13386 90.713699   ::1         ::1           HTTP           555 GET /getUserLocSensorData HTTP/1.1
13389 90.715859   ::1         ::1           HTTP           545 GET /getLocData HTTP/1.1

> Frame 13380: 185 bytes on wire (1480 bits), 185 bytes captured (1480 bits) on interface \Device\NPF_{B2BB52DF-DD75-41E3-AE5B-912308D125CE}, id 0
> Null/Loopback
> Internet Protocol Version 6, Src: ::1, Dst: ::1
> Transmission Control Protocol, Src Port: 8080, Dst Port: 59533, Seq: 2818288, Ack: 30664, Len: 121
v Hypertext Transfer Protocol
  v HTTP/1.1 200 \r\n
    v [Expert Info (Chat/Sequence): HTTP/1.1 200 \r\n]
        [HTTP/1.1 200 \r\n]
        [Severity level: Chat]
        [Group: Sequence]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
  v Content-Length: 0\r\n
      [Content length: 0]
    Date: Thu, 16 Jan 2020 11:34:09 GMT\r\n
    Keep-Alive: timeout=60\r\n
    Connection: keep-alive\r\n
    \r\n
    [HTTP response 67/100]
    [Time since request: 0.009182000 seconds]
    [Prev request in frame: 13362]
    [Prev response in frame: 13366]
    [Request in frame: 13374]
    [Next request in frame: 13386]
    [Next response in frame: 13400]
    [Request URI: http://localhost:8080/setKidsData]
```

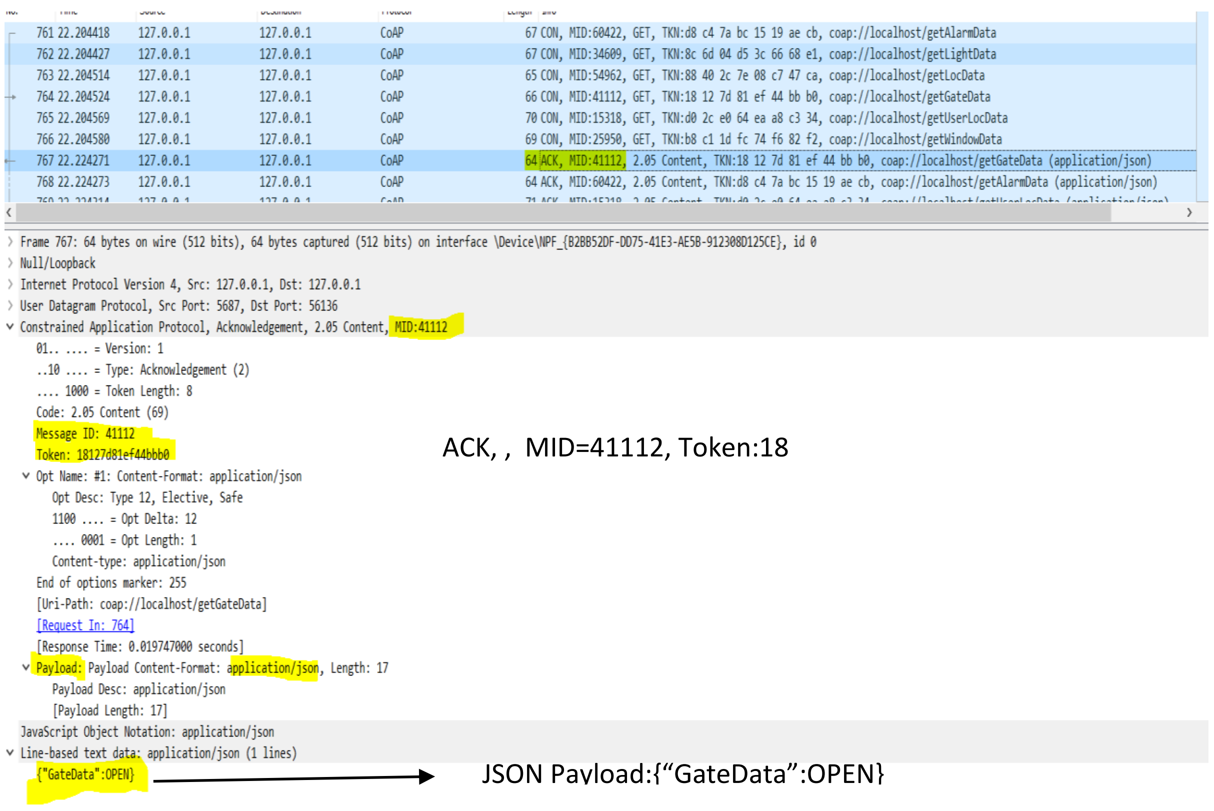
Fig: HTTP POST Response



```
761 22.204418   127.0.0.1   127.0.0.1   CoAP   67 CON, MID:60422, GET, TKN:d8 c4 7a bc 15 19 ae cb, coap://localhost/getAlarmData
762 22.204427   127.0.0.1   127.0.0.1   CoAP   67 CON, MID:34609, GET, TKN:8c 6d 04 d5 3c 66 68 e1, coap://localhost/getLightData
763 22.204514   127.0.0.1   127.0.0.1   CoAP   65 CON, MID:54962, GET, TKN:88 40 2c 7e 08 c7 47 ca, coap://localhost/getLocData
764 22.204524   127.0.0.1   127.0.0.1   CoAP   66 CON, MID:41112, GET, TKN:18 12 7d 81 ef 44 bb b0, coap://localhost/getGateData
765 22.204569   127.0.0.1   127.0.0.1   CoAP   70 CON, MID:15318, GET, TKN:d0 2c e0 64 ea a8 c3 34, coap://localhost/getUserLocData
766 22.204580   127.0.0.1   127.0.0.1   CoAP   69 CON, MID:25950, GET, TKN:b8 c1 1d fc 74 f6 82 f2, coap://localhost/getWindowData
767 22.224271   127.0.0.1   127.0.0.1   CoAP   64 ACK, MID:41112, 2.05 Content, TKN:18 12 7d 81 ef 44 bb b0, coap://localhost/getGateData (application/json)
768 22.224273   127.0.0.1   127.0.0.1   CoAP   64 ACK, MID:60422, 2.05 Content, TKN:d8 c4 7a bc 15 19 ae cb, coap://localhost/getAlarmData (application/json)
769 22.224314   127.0.0.1   127.0.0.1   CoAP   71 ACK, MID:15318, 2.05 Content, TKN:d0 2c e0 64 ea a8 c3 34, coap://localhost/getUserLocData (application/json)

> Frame 764: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{B2BB52DF-DD75-41E3-AE5B-912308D125CE}, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 56136, Dst Port: 5687
v Constrained Application Protocol, Confirmable, GET, MID:41112
    01.. .... = Version: 1
    ..00 .... = Type: Confirmable (0)
    .... 1000 = Token Length: 8
    Code: GET (1)
    Message ID: 41112
    Token: 18127d81ef44bbb0
  v Opt Name: #1: Uri-Host: localhost
      Opt Desc: Type 3, Critical, Unsafe
      0011 .... = Opt Delta: 3
      .... 1001 = Opt Length: 9
      Uri-Host: localhost
  v Opt Name: #2: Uri-Path: getGateData
      Opt Desc: Type 11, Critical, Unsafe
      1000 .... = Opt Delta: 8
      .... 1011 = Opt Length: 11
      Uri-Path: getGateData
    [Uri-Path: coap://localhost/getGateData]
    [Response In: 767]
```

CON, MID=41112, Token:18

Fig: Coap Get Request

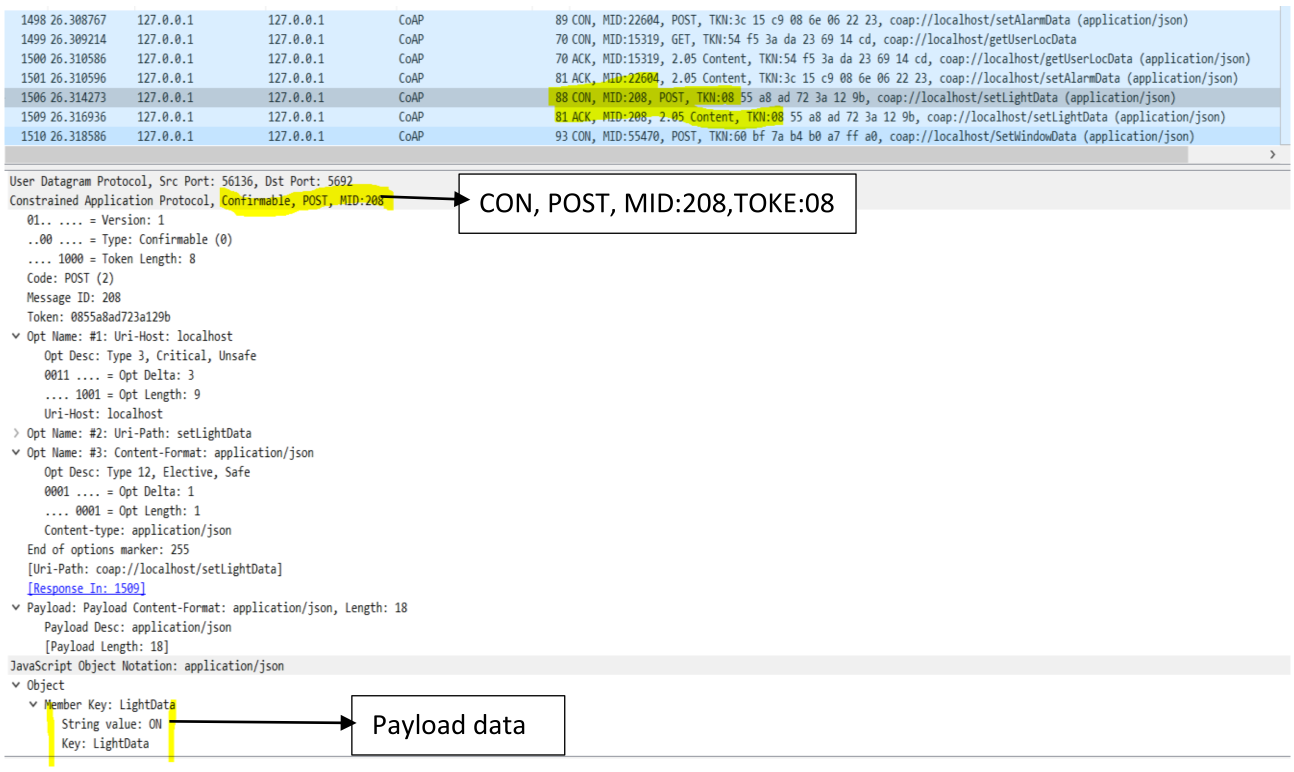


```
761 22.204418   127.0.0.1   127.0.0.1   CoAP   67 CON, MID:60422, GET, TKN:d8 c4 7a bc 15 19 ae cb, coap://localhost/getAlarmData
762 22.204427   127.0.0.1   127.0.0.1   CoAP   67 CON, MID:34609, GET, TKN:8c 6d 04 d5 3c 66 68 e1, coap://localhost/getLightData
763 22.204514   127.0.0.1   127.0.0.1   CoAP   65 CON, MID:54962, GET, TKN:88 40 2c 7e 08 c7 47 ca, coap://localhost/getLocData
764 22.204524   127.0.0.1   127.0.0.1   CoAP   66 CON, MID:41112, GET, TKN:18 12 7d 81 ef 44 bb b0, coap://localhost/getGateData
765 22.204569   127.0.0.1   127.0.0.1   CoAP   70 CON, MID:15318, GET, TKN:d0 2c e0 64 ea a8 c3 34, coap://localhost/getUserLocData
766 22.204580   127.0.0.1   127.0.0.1   CoAP   69 CON, MID:25950, GET, TKN:b8 c1 1d fc 74 f6 82 f2, coap://localhost/getWindowData
767 22.224271   127.0.0.1   127.0.0.1   CoAP   64 ACK, MID:41112, 2.05 Content, TKN:18 12 7d 81 ef 44 bb b0, coap://localhost/getGateData (application/json)
768 22.224273   127.0.0.1   127.0.0.1   CoAP   64 ACK, MID:60422, 2.05 Content, TKN:d8 c4 7a bc 15 19 ae cb, coap://localhost/getAlarmData (application/json)

> Frame 767: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface \Device\NPF_{B2BB52DF-DD75-41E3-AE5B-912308D125CE}, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 5687, Dst Port: 56136
v Constrained Application Protocol, Acknowledgement, 2.05 Content, MID:41112
    01.. .... = Version: 1
    ..10 .... = Type: Acknowledgement (2)
    .... 1000 = Token Length: 8
    Code: 2.05 Content (69)
    Message ID: 41112
    Token: 18127d81ef44bbb0
  v Opt Name: #1: Content-Format: application/json
      Opt Desc: Type 12, Elective, Safe
      1100 .... = Opt Delta: 12
      .... 0001 = Opt Length: 1
      Content-type: application/json
    End of options marker: 255
    [Uri-Path: coap://localhost/getGateData]
    [Request In: 764]
    [Response Time: 0.019747000 seconds]
  v Payload: Payload Content-Format: application/json, Length: 17
      Payload Desc: application/json
      [Payload Length: 17]
  JavaScript Object Notation: application/json
v Line-based text data: application/json (1 lines)
    {"GateData":OPEN}
```

ACK, , MID=41112, Token:18

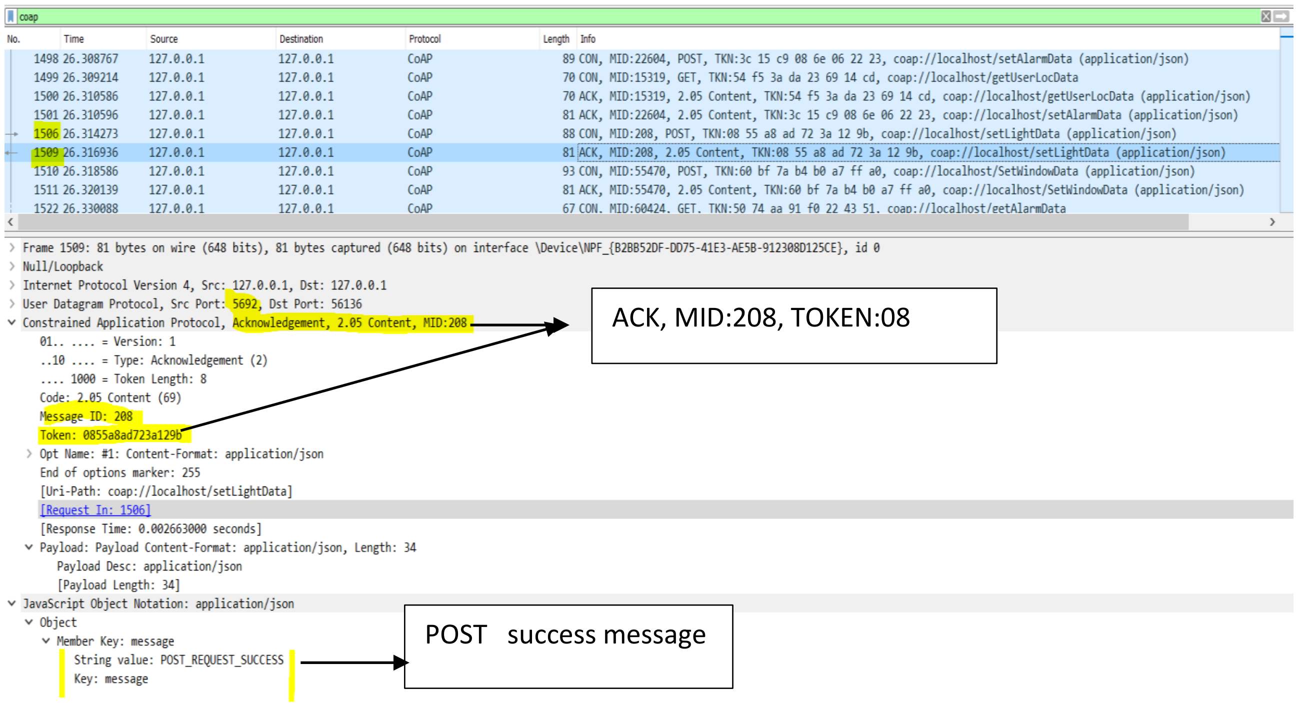
JSON Payload:{“GateData”:OPEN}

Fig: Coap Response



```
1498 26.308767   127.0.0.1   127.0.0.1   CoAP   89 CON, MID:22604, POST, TKN:3c 15 c9 08 6e 06 22 23, coap://localhost/setAlarmData (application/json)
1499 26.309214   127.0.0.1   127.0.0.1   CoAP   70 CON, MID:15319, GET, TKN:54 f5 3a da 23 69 14 cd, coap://localhost/getUserLocData
1500 26.310586   127.0.0.1   127.0.0.1   CoAP   70 ACK, MID:15319, 2.05 Content, TKN:54 f5 3a da 23 69 14 cd, coap://localhost/getUserLocData (application/json)
1501 26.310596   127.0.0.1   127.0.0.1   CoAP   81 ACK, MID:22604, 2.05 Content, TKN:3c 15 c9 08 6e 06 22 23, coap://localhost/setAlarmData (application/json)
1506 26.314273   127.0.0.1   127.0.0.1   CoAP   88 CON, MID:208, POST, TKN:08 55 a8 ad 72 3a 12 9b, coap://localhost/setLightData (application/json)
1509 26.316936   127.0.0.1   127.0.0.1   CoAP   81 ACK, MID:208, 2.05 Content, TKN:08 55 a8 ad 72 3a 12 9b, coap://localhost/setLightData (application/json)
1510 26.318586   127.0.0.1   127.0.0.1   CoAP   93 CON, MID:55470, POST, TKN:60 bf 7a b4 b0 a7 ff a0, coap://localhost/SetWindowData (application/json)

User Datagram Protocol, Src Port: 56136, Dst Port: 5692
Constrained Application Protocol, Confirmable, POST, MID:208
  01.. .... = Version: 1
  ..00 .... = Type: Confirmable (0)
  .... 1000 = Token Length: 8
  Code: POST (2)
  Message ID: 208
  Token: 0855a8ad723a129b
v Opt Name: #1: Uri-Host: localhost
    Opt Desc: Type 3, Critical, Unsafe
    0011 .... = Opt Delta: 3
    .... 1001 = Opt Length: 9
    Uri-Host: localhost
> Opt Name: #2: Uri-Path: setLightData
v Opt Name: #3: Content-Format: application/json
    Opt Desc: Type 12, Elective, Safe
    0001 .... = Opt Delta: 1
    .... 0001 = Opt Length: 1
    Content-type: application/json
  End of options marker: 255
  [Uri-Path: coap://localhost/setLightData]
  [Response In: 1509]
v Payload: Payload Content-Format: application/json, Length: 18
    Payload Desc: application/json
    [Payload Length: 18]
JavaScript Object Notation: application/json
v Object
  v Member Key: LightData
      String value: ON
      Key: LightData
```

Fig: coap Post request



```
coap
No.  Time        Source      Destination  Protocol  Length Info
1498 26.308767   127.0.0.1   127.0.0.1    CoAP      89 CON, MID:22604, POST, TKN:3c 15 c9 08 6e 06 22 23, coap://localhost/setAlarmData (application/json)
1499 26.309214   127.0.0.1   127.0.0.1    CoAP      70 CON, MID:15319, GET, TKN:54 f5 3a da 23 69 14 cd, coap://localhost/getUserLocData
1500 26.310586   127.0.0.1   127.0.0.1    CoAP      70 ACK, MID:15319, 2.05 Content, TKN:54 f5 3a da 23 69 14 cd, coap://localhost/getUserLocData (application/json)
1501 26.310596   127.0.0.1   127.0.0.1    CoAP      81 ACK, MID:22604, 2.05 Content, TKN:3c 15 c9 08 6e 06 22 23, coap://localhost/setAlarmData (application/json)
1506 26.314273   127.0.0.1   127.0.0.1    CoAP      88 CON, MID:208, POST, TKN:08 55 a8 ad 72 3a 12 9b, coap://localhost/setLightData (application/json)
1509 26.316936   127.0.0.1   127.0.0.1    CoAP      81 ACK, MID:208, 2.05 Content, TKN:08 55 a8 ad 72 3a 12 9b, coap://localhost/setLightData (application/json)
1510 26.318586   127.0.0.1   127.0.0.1    CoAP      93 CON, MID:55470, POST, TKN:60 bf 7a b4 b0 a7 ff a0, coap://localhost/SetWindowData (application/json)
1511 26.320139   127.0.0.1   127.0.0.1    CoAP      81 ACK, MID:55470, 2.05 Content, TKN:60 bf 7a b4 b0 a7 ff a0, coap://localhost/SetWindowData (application/json)
1522 26.330088   127.0.0.1   127.0.0.1    CoAP      67 CON, MID:60424, GET, TKN:50 74 aa 91 f0 22 43 51, coap://localhost/getAlarmData

> Frame 1509: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface \Device\NPF_{B2BB52DF-DD75-41E3-AE5B-912308D125CE}, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 5692, Dst Port: 56136
v Constrained Application Protocol, Acknowledgement, 2.05 Content, MID:208
    01.. .... = Version: 1
    ..10 .... = Type: Acknowledgement (2)
    .... 1000 = Token Length: 8
    Code: 2.05 Content (69)
    Message ID: 208
    Token: 0855a8ad723a129b
  > Opt Name: #1: Content-Format: application/json
    End of options marker: 255
    [Uri-Path: coap://localhost/setLightData]
    [Request In: 1506]
    [Response Time: 0.002663000 seconds]
  v Payload: Payload Content-Format: application/json, Length: 34
      Payload Desc: application/json
      [Payload Length: 34]
v JavaScript Object Notation: application/json
  v Object
    v Member Key: message
        String value: POST_REQUEST_SUCCESS
        Key: message
```

Fig: caop post response

8. URI Summary

8.1 CoAP URI:-

Coap URI	Name of Resource(sensor/actuator)
coap://localhost:5685/getLocData	Dog Position sensor
coap://localhost:5686/getAlarmData	Alarm sensor
coap://localhost:5687/getGateData	Gate Sensor
coap://localhost:5688/setAlarmData	Alarm actuator
coap://localhost:5689/setGateData	Gate actuator
coap://localhost:5690/getUserLocData	User Location sensor
coap://localhost:5691/getLightData	Light Sensor
coap://localhost:5692/setLightData	Light actuator
coap://localhost:5693/getWindowData	Window sensor
coap://localhost:5694/SetWindowData	Window actuator
coap://localhost:6000/getKidsLocData	Kids Location sensor
coap://localhost:6001/setKidsData	Kids safety actuator
coap://localhost:6002/getKidsAlert	Kids Alert sensor
coap://localhost:6003/setAlertData	Kids Alert actuator

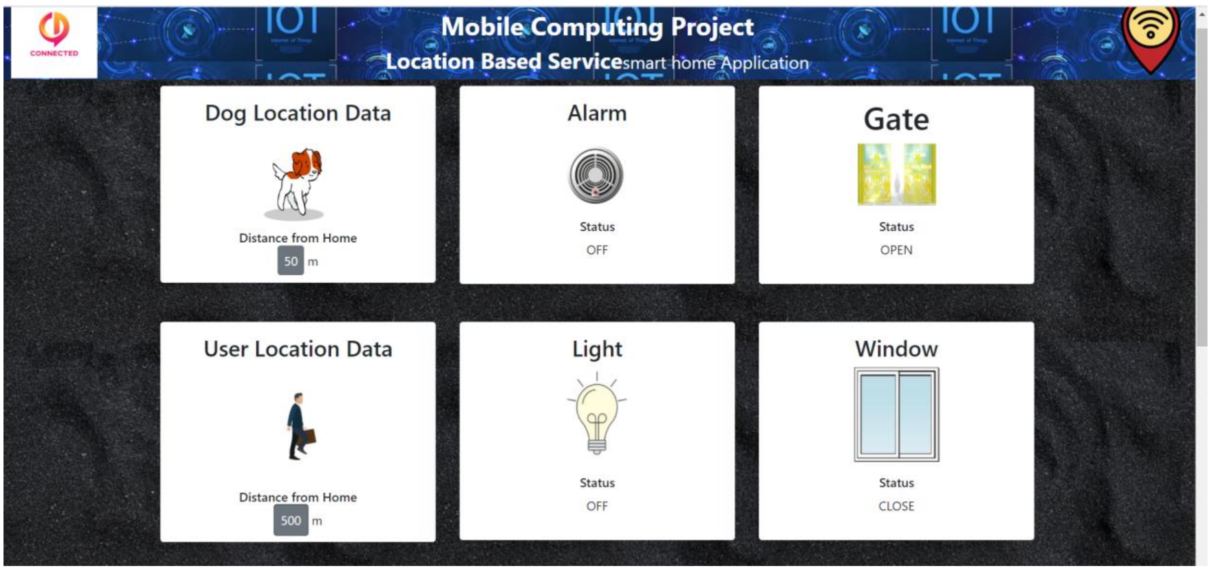
8.2 Http URI

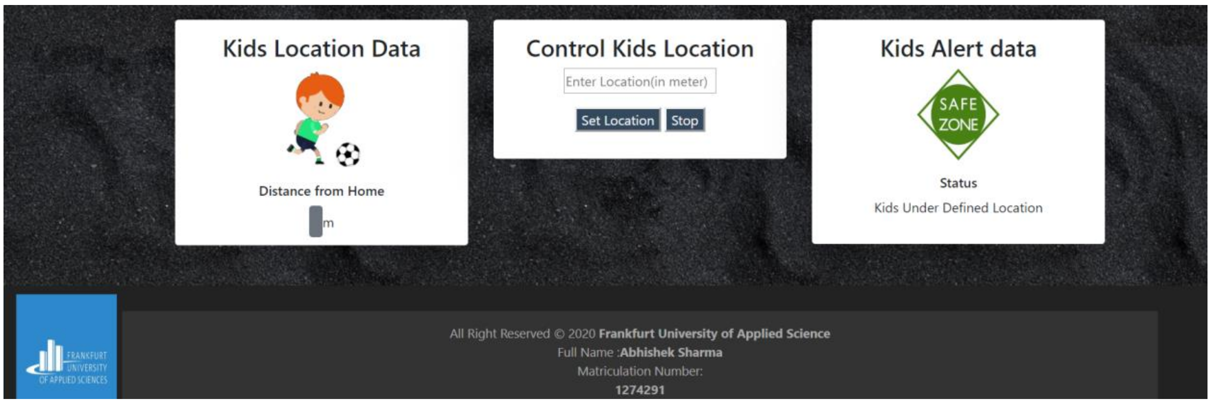
HTTP URI	Summary in short
http://localhost:8080/getLocData	GET(return dog location Numeric data)
http://localhost:8080/getAlarmData	GET(return Alarm data ON/OFF)
http://localhost:8080/getGateData	GET(return Gate data OPEN/CLOSE)
http://localhost:8080/getUserLocSensorData	GET(return user location Numeric data)
http://localhost:8080/getLightData	GET(return Light Data ON/OFF)
http://localhost:8080/getWindowData	GET(return Window data OPEN/CLOSE)
http://localhost:8080/getKidsData	GET(return kids location Numeric data)
http://localhost:8080/getKidsAlert	GET(return message Kids under defined location /kids out of range)
http://localhost:8080/setKidsData	POST(user input entered location data)

9 . Manual to run Program:

1. Open CMD and traverse up to project location
2. Enter mvnw clean
3. Enter mvnw install- Jar file will be created in Target repository
   And enter mvnw spring-boot:run then follow step 7.
4. Copy and paste jar file to some other location
5. Open CMD again where jar file is saved
6. Enter Java -jar <jar file name>
7. Open browser and enter http://localhost:8080

   Now below console will appear





All Right Reserved © 2020 **Frankfurt University of Applied Science**
Full Name :**Abhishek Sharma**
Matriculation Number:
1274291

Note :for Use-case 3(Kids Location data)- First enter any value in input box(Enter location in meter)then only hit the button Set Location.

*This HTML page is validated using w3 school validator

10. References

[1] https://tools.ietf.org/html/rfc7252

[2] http://hinrg.cs.jhu.edu/joomla/images/stories/coap-ipsn.pdf

[3] https://www.w3schools.in/http-tutorial/intro/

[4] https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

[5] [Shelby11] Zach Shelby, "CoRE Link Format," draft-ietf-core-link- format-07 https://tools.ietf.org/html/draft-ietf-core-link-format-01

[6] [Z. Shelby13] Z. Shelby, Sensinode, K. Hartke, "Constrained Application Protocol (CoAP)," draft-ietf-core-coap-18. http://tools.ietf.org/html/draft-ietf-core-coap-18