REMOTE HOME ACCESS USING AMAZON WEB SERVICES

A PROJECT REPORT

Submitted by

REVATHI.G. (312215106089)
R. PRIYADARSHINI (312215106091)
SHASHANK KARRTHIKEYAA.A.S. (312215106102)

in the partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING
ANNA UNIVERSITY: CHENNAI 600025

APRIL 2019

ANNA UNIVERSITY: CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report **"REMOTE HOME ACCESS USING AMAZON WEB SERVICES"** is the bonafide work of "**REVATHI.G., R. PRIYADARSHINI, SHASHANK KARRTHIKEYAA.A.S.**" who carried out the project under my supervision.

SIGNATURE
Dr.S.Radha
HEAD OF THE DEPARTMENT

Department of Electronics and Communication Engineering,
Sri Sivasubramaniya Nadar College of Engineering,
Kalavakkam,
Chennai-603110.

SIGNATURE
Dr.S.Sakthivel Murugan
SUPERVISOR
Associate Professor,
Department of Electronics and Communication Engineering,
Sri Sivasubramaniya Nadar College of Engineering,
Kalavakkam,
Chennai-603110.

Submitted for the examination held on…….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We would like to express our deepest appreciation to all those who presided us with the possibility to complete this project. We are highly indebted to SSN COLLEGE OF ENGINEERING and our Principal, **Dr. S. Salivahanan** for their guidance and support by providing the facilities to carry out this project.

We acknowledge with sense of gratitude, our sincere thanks to our beloved guide **Dr. S. Radha**, Professor and Head of the Department, Electronics and Communication Engineering, for her invaluable guidance, innovative suggestions, patience and unflinching support throughout the course of this project.

We would like to also like to extend our sincere and wholehearted thanks to our panel coordinators **Dr. R. Jayaparvathy**, Professor, Department of Electronics and Communication Engineering, and **Dr. I. Nelson**, Assistant Professor, Department of Electronics and Communication Engineering whose contribution in stimulating suggestions and encouragement, helped us to coordinate throughout the project work.

We would like to extend our sincere gratitude and thanks to **Mr. S. Vijay Anand**, Assistant Vice President (Technology), Aricent Inc. for his valuable inputs, guidance and time for mentoring this project throughout its successful device testing in Aricent and implementation in our Under Water Acoustic Research Lab.

Finally, with great enthusiasm, we express our thanks to all our department faculty and technical staff, friends and parents for their sustained support and interest in the completion of our project.

TABLE OF CONTENTS

ABSTRACT

Internet of Things (IoT) is one of the most swiftly growing technologies, becoming more and more present, creating a profound impact on the quality of living with each passing day. It encircles newer and better technologies such as smart cities, smart electric grids, Remote smart homes etc. These technologies are developed from the existing systems, which are integrated with the physical world to create much better ones. Remote-control services for smart home have become quite popular now-a-days where the home user outside of his/her home can remotely monitor and control home devices. This project displays a system, where the requests generated by user within the home or outside the home can be intelligently managed and controlled by Cloud based Services (AWS), using high level functionalities like auto-discovery; auto-detection of devices; auto-transmission of devices data; auto-switching tasks and so on and so forth. The functions defined for this Remote Home access (RHA) system can be classified into various service groups such as **Monitoring, Controlling, Reporting, Authenticating, Authorizing, Managing, logging and Temporary user groups.** As part of the RHA architecture, the connectivity between the Cloud and the user application is encrypted using Secure Socket Layer (SSL) certificates so that the infringement of user privacy is prevented.

Keywords: Internet of Things, RHA, SSL.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

S.No	Abbrevation	Caption
1	REST	Representative State Transfer
2	API	Application Programming Interface
3	HTTPS	Hyper Text Transfer Protocol Secure
4	SQL	Structure Query Language
5	Node MCU	Node Micro Controller Unit
6	AC	Alternating Current
7	DC	Direct Current
8	V	Volts
9	mA	Milli Ampere
10	ms	Milli Second

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION: PROJECT OVERVIEW AND OBJECTIVES

Internet of Things (IoT) finds its application across all kinds of markets and user groups. Smart homes or automated homes is one such part of the Internet of Things which constitute a group of Wi-Fi enabled devices that can regulate themselves according to user priorities and are controlled by a user interface(app), providing comfort and convenience to the people who own them. Such a system can make all the difference in the quality of our lives not only in terms of comfort and convenience but also in terms of the bills we pay. This project aims to develop such a smart home system, which could be accessed from a remote space using Amazon Web Services (AWS) at the backend.

1.2 REMOTE HOME ACCESS:

Remote Home Access is an integral part of IoT which aims to customize our personal abodes making it more optimized, efficient and easily accessible. Currently, a lot of smart home devices are available in the market, but all of them are derived from and are made to support a specific vendor. Hence, the aim of this project is to create a general interface that could be customized later on, according to the requirements of the user, that is, to develop an android application (User interface) based Remote Home Automation system using Amazon Web Services (AWS) as the backend interface with multiple in-built functionalities.

1.3 REMOTE HOME ACCESS FEATURES:

1) MONITORING: Switch status (ON/OFF) or the status of the devices such as lights or blinds is monitored via the AWS cloud, gateway, and the mobile application.

2) CONTROLLING: Switch status (ON/OFF) or the status of the devices such as lights or blinds is controlled via the AWS cloud, gateway, and the mobile application.
3) AUTHENTICATING: Login and user authentication is performed to control and provide system access to users.
4) AUTHORIZING: Users and system functions are divided into some groups and access rights of the users are controlled according to the groups of users or functions.
5) MANAGING: User addition and deletion, change of password and change of other system parameters such as port number etc.
6) LOGGING: Home user login, device control/monitoring, user addition/deletion and other parameters are logged in files in terms of occurrence time, subject and event description.

a. SECURITY: Secure Socket Layer (SSL) **[6]** certificates are used to encrypt the application to server connectivity, serving as a basic layer of security to take care of user privacy and data safety.

1.4 BLOCK DIAGRAM:

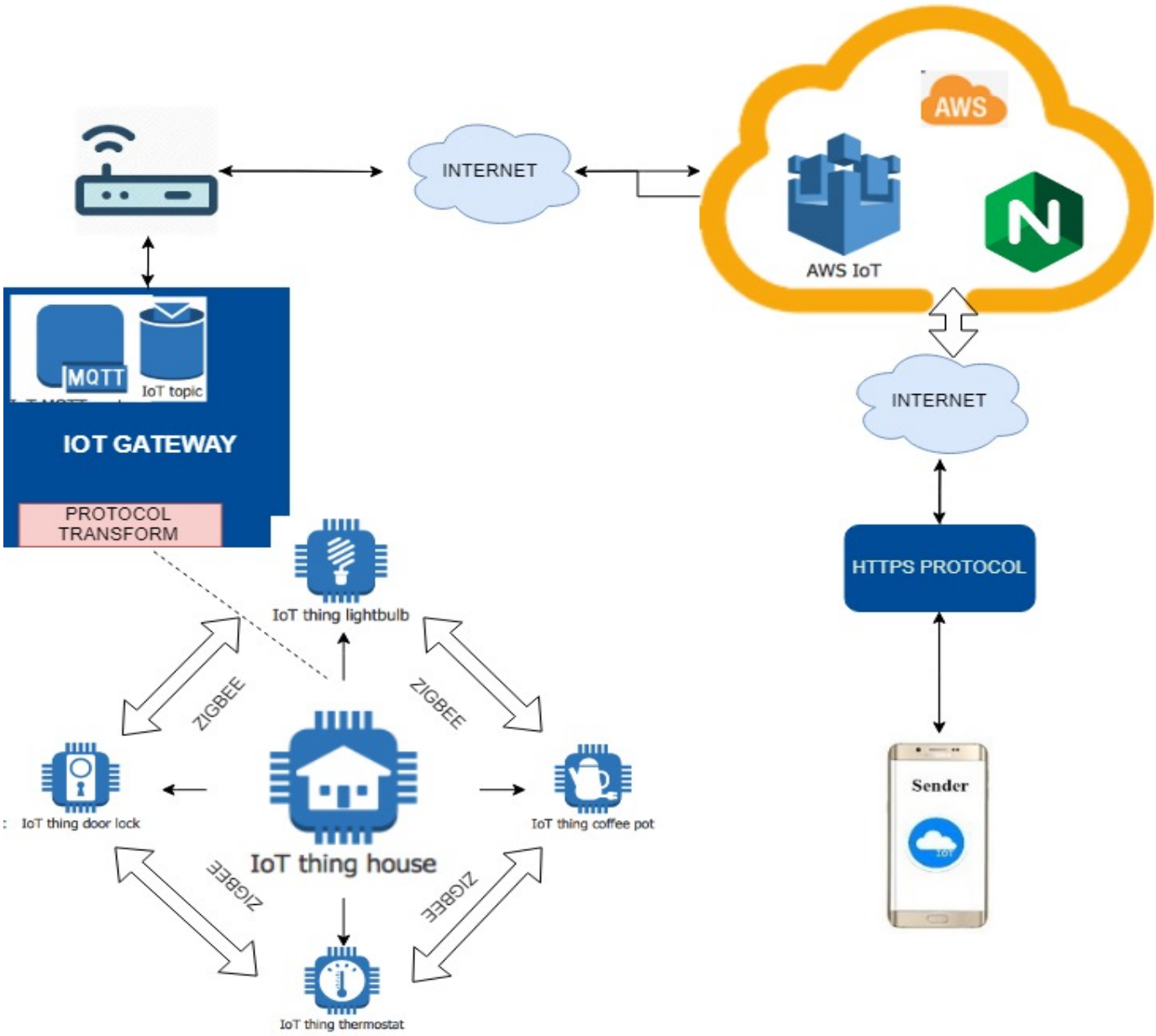

Fig. 1.1 Block Diagram

1.5 CONSTITUENTS:

1) AMAZON WEB SERVICES (AWS):

   Amazon Web Services provides an on-demand cloud computing platform to an individual or a group of individuals. Here we use AWS cloud as the base platform on which the production server (NGNIX) and the MQTT Broker (VerneMQ) are run. The database service availed is SQLite3, the API is a custom-built REST API constructed using flask.

2) NGNIX: It is a free, open source software that is a web server. It can be deployed to serve dynamic HTTP content WSGI application server on the network and it also serves as a software load balancer. Its mail proxy feature provides SSL support. In this project, it basically serves as the production server environment.

3) VerneMQ: It is a high-performance MQTT broker which scales both horizontally and vertically to support a high number of concurrent publishers and subscribers ensuring low latency, reliability and fault tolerance at the same time.

4) ANDROID APPLICATION:

   A user interface for Android OS is developed using the Android studio. Android Studio is an Integrated Development Environment (IDE) powered by IntelliJ IDEA software. The HTTPS services are availed by importing android HTTPS libraries. The HTTP requests are secured by adding self-signed SSL certificates generated using Open SSL, to the android trusted key store.

5) IOT GATEWAY:

In this project the IOT devices are directly connected to NODE MCUs via a relay and the MQTT Client is directly run on the NODE MCU without any intermediate gateway. This can be further extended to a setup where devices are connected to a pi gateway in case a large number of devices or big sophisticated devices are used or when different protocols are used to control different group of devices.

1.6 PROTOCOLS USED:

1) Hyper Text Transfer Protocol Secure (HTTPS): It is an extension of the HTTP used for secure communication of data over a computer network. The protocol is encrypted using SSL certificates to ensure data integrity. In this project, the HTTPS protocol is used for communication between the Android application and the AWS server.
2) Message Queuing Telemetry Transport (MQTT): It is a publish-subscribe based message transfer protocol. It is a protocol primarily used for accessing remote locations since it uses a small code footprint to transfer data and hence the packet overhead is limited leading to limited bandwidth utilization, making it suitable for long distance data transmission. In this project, the MQTT protocol is used for data transmission between the IoT devices (MQTT clients) and the MQTT broker (VerneMQ in AWS). This connectivity is also encrypted using SSL certificates to ensure a round trip secure data transfer.

1.7 OBJECTIVE:

The aim of this project is to develop an android app based smart Remote Home Access (RHA) system using Amazon Web Services (AWS) as back end interface, integrated with high level functionalities such as Authentication, Authorization, Managing, Monitoring, Controlling, Logging and security and to

create a platform which can serve as a general interface that could later be customized according to the needs of the user or a group of users.

1.8 LITERATURE SURVEY:

Remote Home Access (RHA) and smart home automation systems are the most common and present technologies of today's buzz word IoT (Internet of Things) and the following research works have given us some good insights on various concepts involved and served as an aid for us to dig deeper into our domain (Remote Home access, security, and other related areas.)

1) **Dongyu Wang and Dixon Lo[1]** proposed home appliances controlling and monitoring system through a platform designed to connect sensor data with users' daily life. The sensor data is being processed by a single board computer and delivered to a mobile application through a wireless connection.
2) **Andreas Jacobsson and Paul Davidsson** introduced a model of security and privacy for smart homes. They have performed a very extensive analysis of the risk associated with the current IoT based smart home systems and have developed security design principles to enable control of the risk exposure.
3) **Kumar Mandula and Ramu Parupalli** throw light on realizing a smart home automation system using a microcontroller and a mobile application. The main feature of this paper is that it has constructed two prototypes, one using Bluetooth in the indoor environment and the other one using Ethernet in an outdoor environment, and hence highlighting the differences between in-home access and remote home access.
4) **Majid Al-Kuwari and Abdulrhman Ramadan** proposes a complete design of an IoT based monitoring system for smart home automation. It uses a EmonCMS cloud server platform for collecting data using Node MCU (ESP8266) which allows real-time data sensing and visualizing monitored data and also remote controlling of devices and appliances.

5) **Pavithra.D and Ranjith Balakrishnan** introduced a system of controlling home appliances through smartphones using Wi-Fi as a communication protocol and Raspberry-pi as a server system. Devices with different loads such as lights, fans, and door locks have been controlled through a web interface in their project. Hence the hardware aspects of this paper proved useful in our implementation of the project.
6) **Shopan Dey and Ayon Roy** focus on developing a smart home automation system where the devices connected to the cloud server are controlled by a single admin which facilitate a number of users to which a number of sensor and control nodes are connected. The admin can access and control all the nodes connected to each user but a single user can control only the nodes to which the user itself is connected giving rise to the concept of authorization.

7) **Mahfuzur Rahman and Prabir Bhattacharya** proposed architecture for secure access to a home or an organization's networks and control of networked appliances inside a home or within an organization from a remote location. They use biometrics features and a one-time password mechanism on top of secure socket layer (SSL) for authentication
8) **Mariana Nikolova and Frans Meijs** describe an approach to interconnect home appliances such as TV, Radio etc, in a home network and access the network using a web pad or a mobile phone and they have also demonstrated its feasibility through a typical prototype based on HAVi and WAP standards.
9) **D. Wang, Y. Murase and K. Sugiura** developed a thesis on which the technology trend was evaluated and sought for new possibilities in appliances controlling environment. The concept of activity-based controlling process was verified, and a feasible model was provided for related system and products.

1.9 CONCLUSION FROM LITERATURE SURVEY:

The various research works facilitated us by providing relevant and worthwhile information about the current trends existing in our domain, the pitfalls faced by the industry and risk analysis and acceptable solutions. Let aside the above facts, they also served as a big inspiration for us to take up our project with full thrust and helped us find some solutions for the problems addressed by them and in helping us addressing them through their very extensive research and pellucid explanations.

1.10 REPORT ORGANIZATION:

- Chapter 2 provides a detailed overview of all the packages and functionalities used at the backend. Its scope includes SQLite3 database, VerneMQ broker, NGINX server, REST API, so on and so forth.
- Chapter 3 provides a broad outline on the user interface created for Android OS using an android studio, explaining about different libraries used and highlighting the features involved in the app.
- Chapter 4 throws light on the hardware aspects of the project, namely the Node MCUs and the devices and explains in detail about the constructed circuitry.
- Chapter 5 illustrates the results or final outcomes of the project such as the integration of android app with the API using HTTPS protocol and the establishment of a successful communication link between the cloud and the devices using MQTT, thus creating a complete link between the user interface and the IoT device via AWS and the integration of IOT bridge along with conclusion

CHAPTER 2
BACKEND DESIGNING

2.1 INTRODUCTION TO BACKEND DESIGNING:

In any application, there may be many layers between hardware and end user. They can be differentiated as Front end and Back end. The front end is an abstraction, simplifying the underlying logic by providing a user-friendly interface, while the backend usually handles business logic and storage. The backend should have the computational power to handle multiple requests from the user, multiple hardware and also to store huge amounts of data. Generally, the backend is developed and tested in a development environment and then hosted on a production environment for real-time usage.

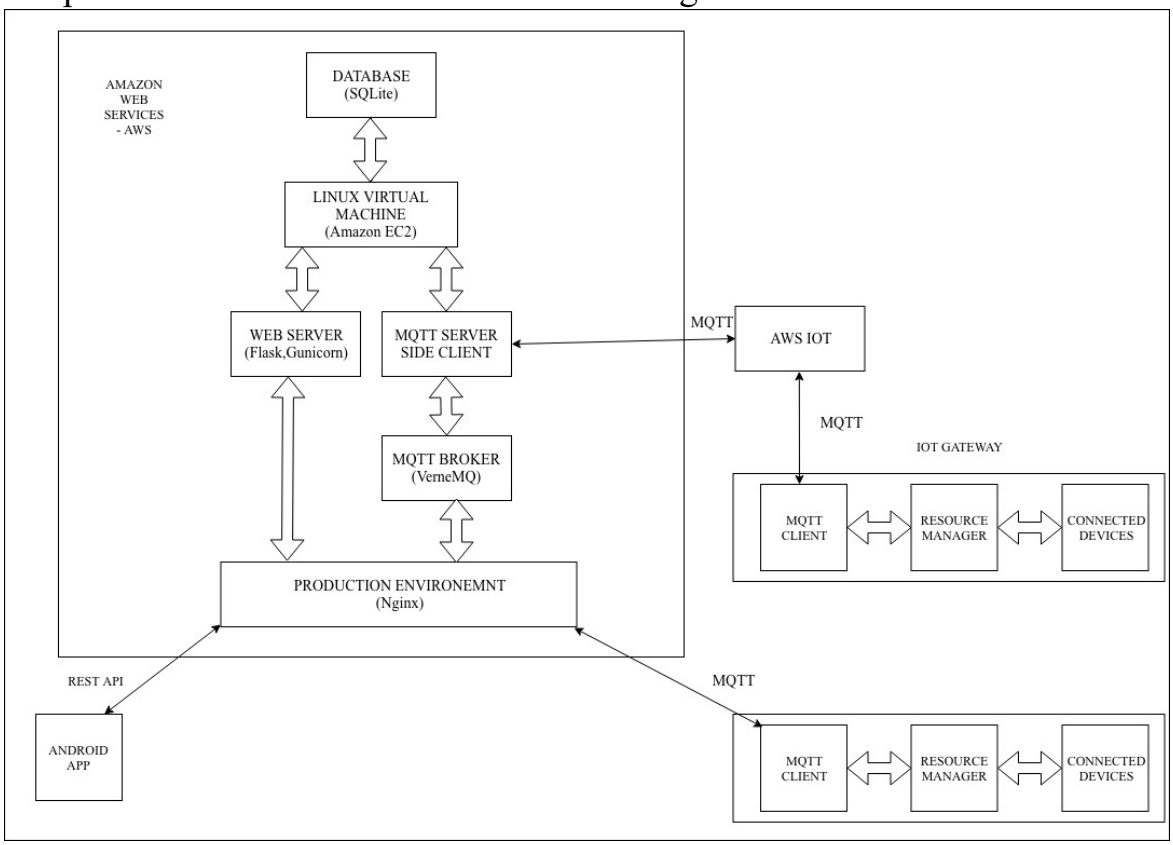

Fig.. 2.1 Flow Diagram

In this application an Open Source Python Framework called as Flask is used for developing REST API's, SQLite is used as a database to store data, VerneMQ an Open Source scalable tool is used as MQTT Broker to handle MQTT communications and NGINX is used as a production server to provide a scalable environment. All communication from the server is encrypted using SSL Certificates. Self-defined algorithms using python are used to provide Authorization, Authentication, Grouping, Logging, Monitoring and Controlling in using the Flask Framework. Extended support is also provided to communicate with AWS IOT to enable communication with devices configured for AWS IOT. The entire backend is hosted on an AWS EC2 platform to cater high usage demands with great reliability and security.

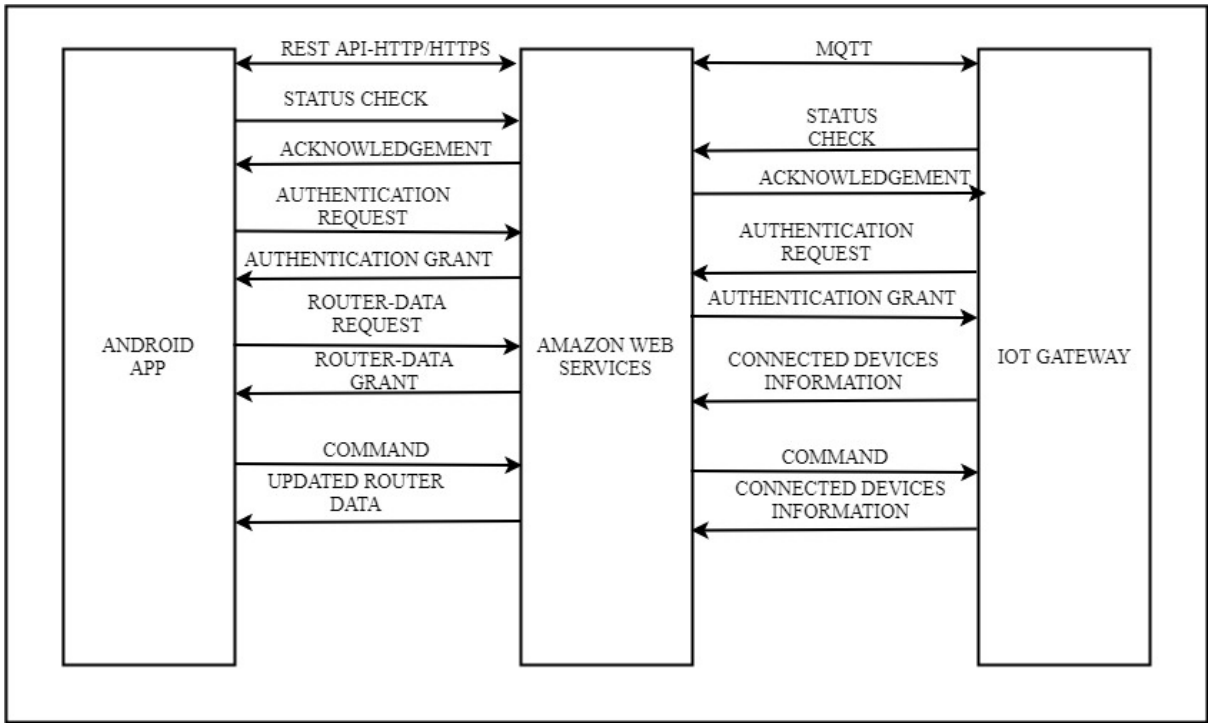

Fig. 2.2 Communication Diagram

The above is the communication diagram for various communications between App, AWS and IOT Gateway. As shown in the diagram there is no direct communication from the App which is using https for communication and the device which is using MQTT for communication. They both interact with the server for operation.

2.2 REST API's:

2.2.1 INTRODUCTION TO REST API's:

Representational State Transfer (REST) is a software architecture that defines a set of rules for creating Web Services. Web services that conform to the REST architecture style are termed as RESTful Web Services. They provide interoperability between computer systems and applications on the Internet.

The following are the constraints in REST Architecture:

1. Client-Server Architecture:

   The principle behind the client-server constraints is the separation of concerns. Separating the user interface concerns from the data storage concerns improves the portability of the user interfaces across multiple platforms. It also improves scalability by simplifying the server components

2. Statelessness:

The client-server communication is constrained by no client context being stored on the server between requests. Each request from any client contains all the information necessary to service the request, and the session state is held in the client. The session state can be transferred by the server to another service such as a database to maintain a persistent state for a period and allow authentication. The client begins sending requests when it is ready to make the transition to a new state. While one or more requests are outstanding, the client is considered to be in transition. The representation of each application state contains links that can be used the next time the client chooses to initiate a new state-transition.

3. Caching ability:

As on the World Wide Web, clients and intermediaries can cache responses. Responses must therefore, implicitly or explicitly, define themselves as cacheable or not to prevent clients from getting stale or inappropriate data in response to further requests. Well-managed caching partially or completely eliminates some client-server interactions, further improving scalability and performance.

4. Layered system:

A client cannot ordinarily tell whether it is connected directly to the end server, or to an intermediary along the way. Intermediary servers can improve system scalability by enabling load balancing and by providing shared caches. They can also enforce security policies.

5. The code on Demand:

Servers can temporarily extend or customize the functionality of a client by transferring executable code: for example, compiled components such as Java applets, or client-side scripts such as JavaScript.

6. Uniform Interface:

The uniform interface constraint is fundamental to the design of any RESTful system. It simplifies and decouples the architecture, which enables each part to evolve independently.

7. Applied to Web Services:

Web service APIs that adhere to the REST architectural constraints are called RESTful API's. HTTP-based RESTful APIs are defined with the following aspects:

- A base URI, such as http://api.example.com/collection/;

- A standard HTTP method (e.g., GET, POST, PUT, PATCH and DELETE);
- A media type that defines state transition data elements

The constraints in the REST architecture style affect the following properties:

1. Performance in Component Interactions which is a key factor in network efficiency and performance.
2. Scalability allowing the support of large numbers of components and interactions among components.
3. The simplicity of the uniform interface.
4. Modifiability of components to meet changing needs.
5. Visibility of communication between components by service agents
6. Portability of components by moving program code with the data
7. Reliability in the system level against failures and malfunctions

The following are the HTTP Methods:

1. Get:

Retrieve the URIs of the member resources of the collection resource in the response body

2. Post:

Create a member resource in the collection resource using the instructions in the request body. The URI of the created member resource is automatically assigned and returned in the response Location header field

3. Put:

Replace all the representations of the member resources of the collection resource with the representation in the request body or create the collection resource if it does not exist.

4. Patch:

Update all the representations of the member resources of the collection resource using the instructions in the request body or may create the collection resource if it does not exist.

5. Delete:

Delete all the representations of the member resources of the collection resource.

2.2.2 INTRODUCTION TO FLASK FRAMEWORK:

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries (except for some basics standard libraries). It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program.

Applications that use the Flask framework include Pinterest, LinkedIn, and the community web page for Flask itself.

2.2.3 CONNECTION FLOW:

a) Endpoint:

The Web service endpoint is the port upon which you connect a Web service client to the server. The Web service client is connected to the Web service's endpoint at the following URL, in which the server is running. The Web service runs over HTTP or HTTPS.

b) Access Token:

An access token contains the security credentials for a login session and identifies the user.

c) Refresh Token:

The refresh token is used to generate a new access token if the previous one has expired.

d) Endpoint Functions:

The following are the endpoints.

a. Sign Up
b. Login
c. Logout
d. Forgot Password
e. Add Router
f. Remove Router
g. Add User
h. Remove User
i. Monitoring
j. Controlling

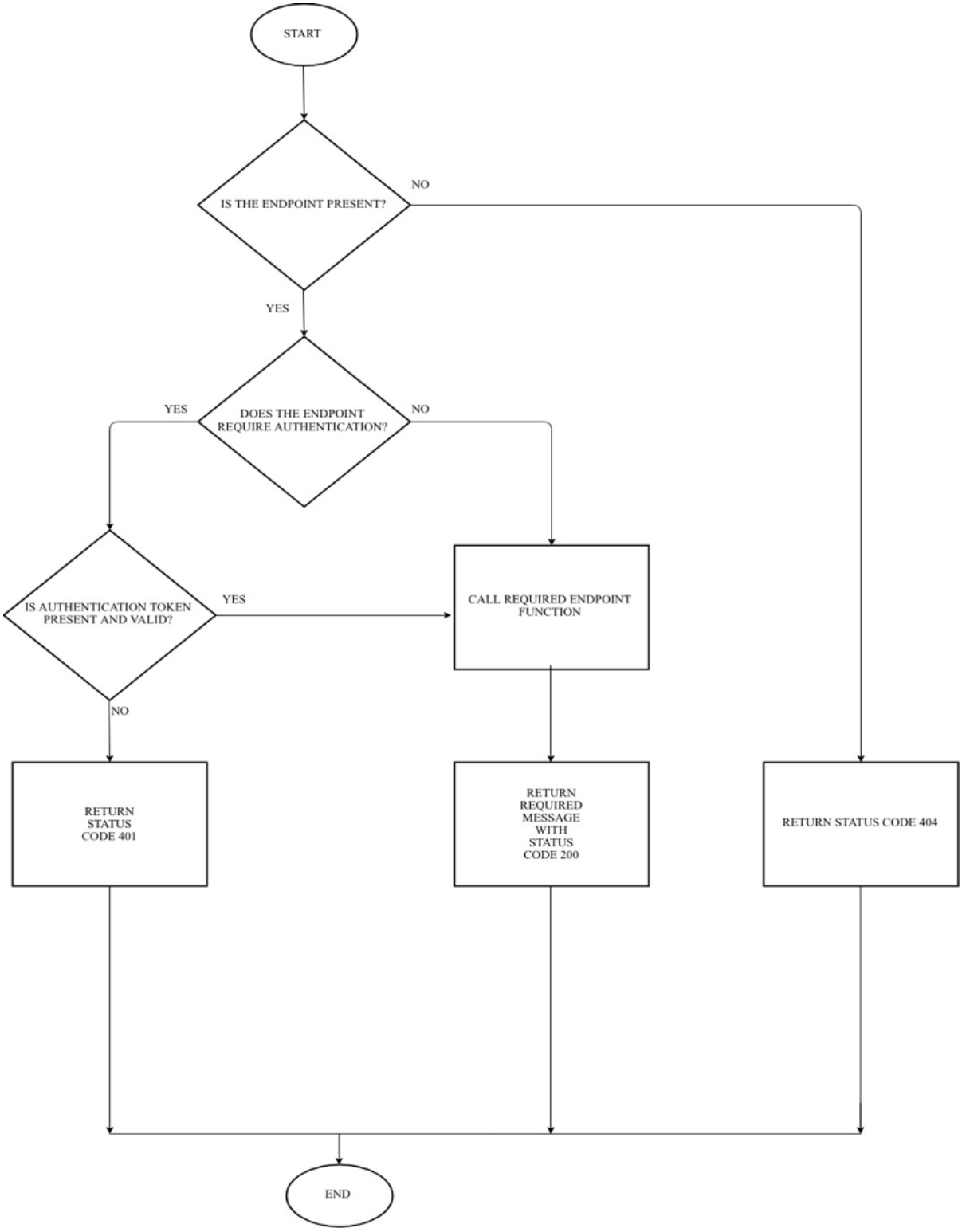

Fig. 2.3 Flask Connection Flow Diagram

a) Sign Up:

In any application involving lot of users Sign Up is a very important option. It allows for the creation of new users.

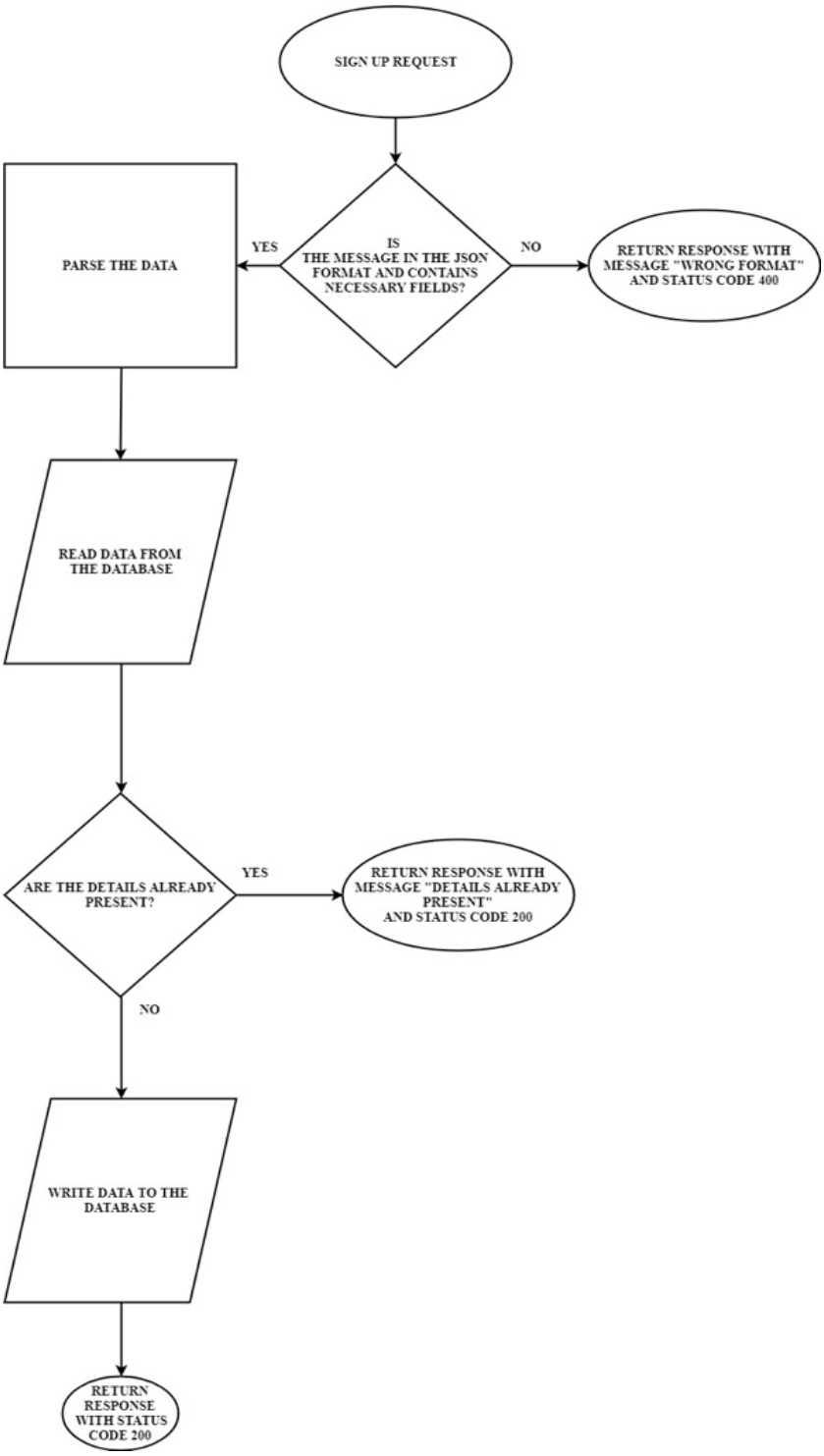

Fig. 2.4 Flow Chart for Sign Up Operation

b) Login:

Login is an essential operation to identify the user and to customize options.

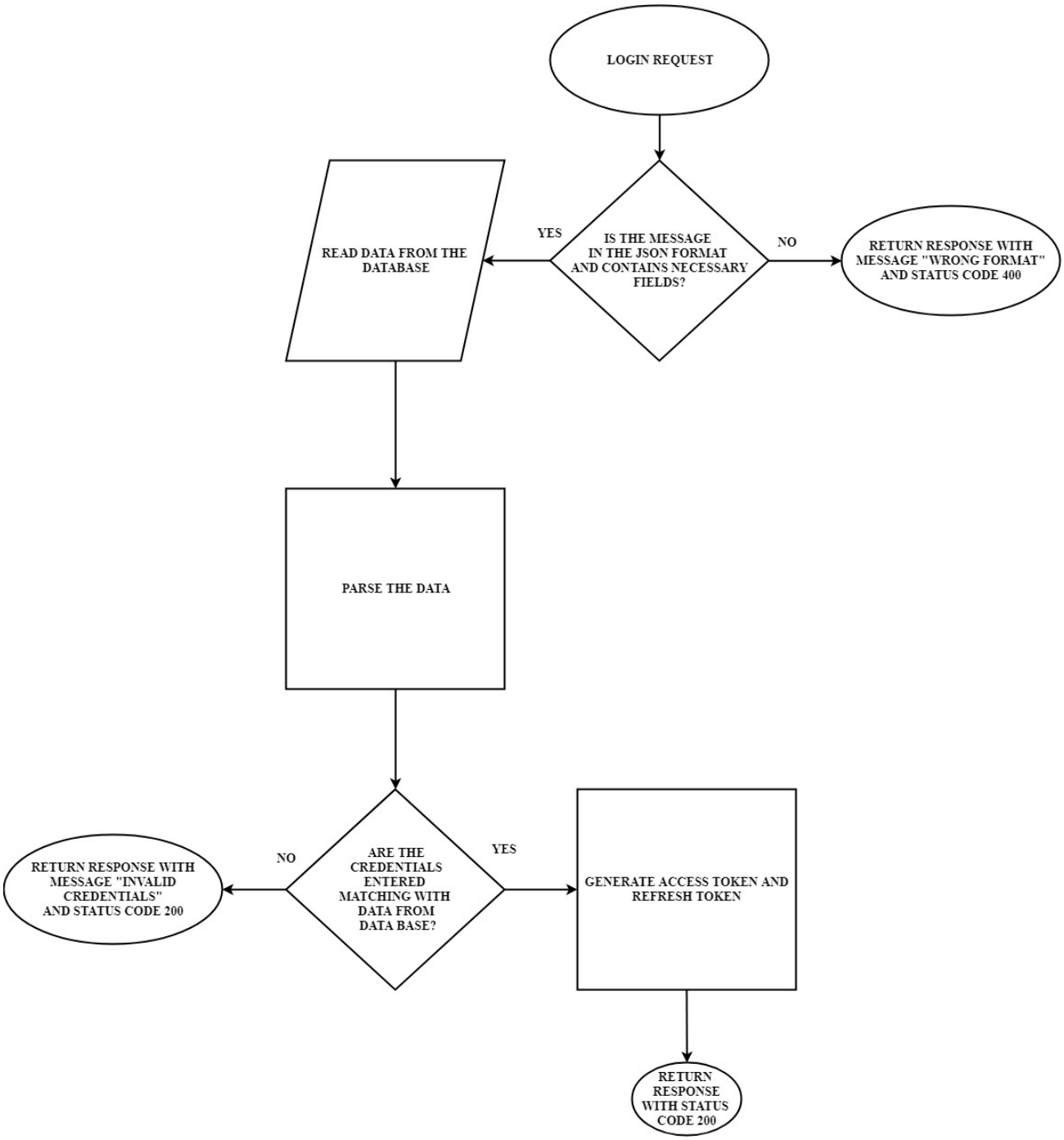

Fig. 2.5 Flow Chart for Login Operation

c) Logout:

This is used to revoke access to the user once he/she has completed his/her job.

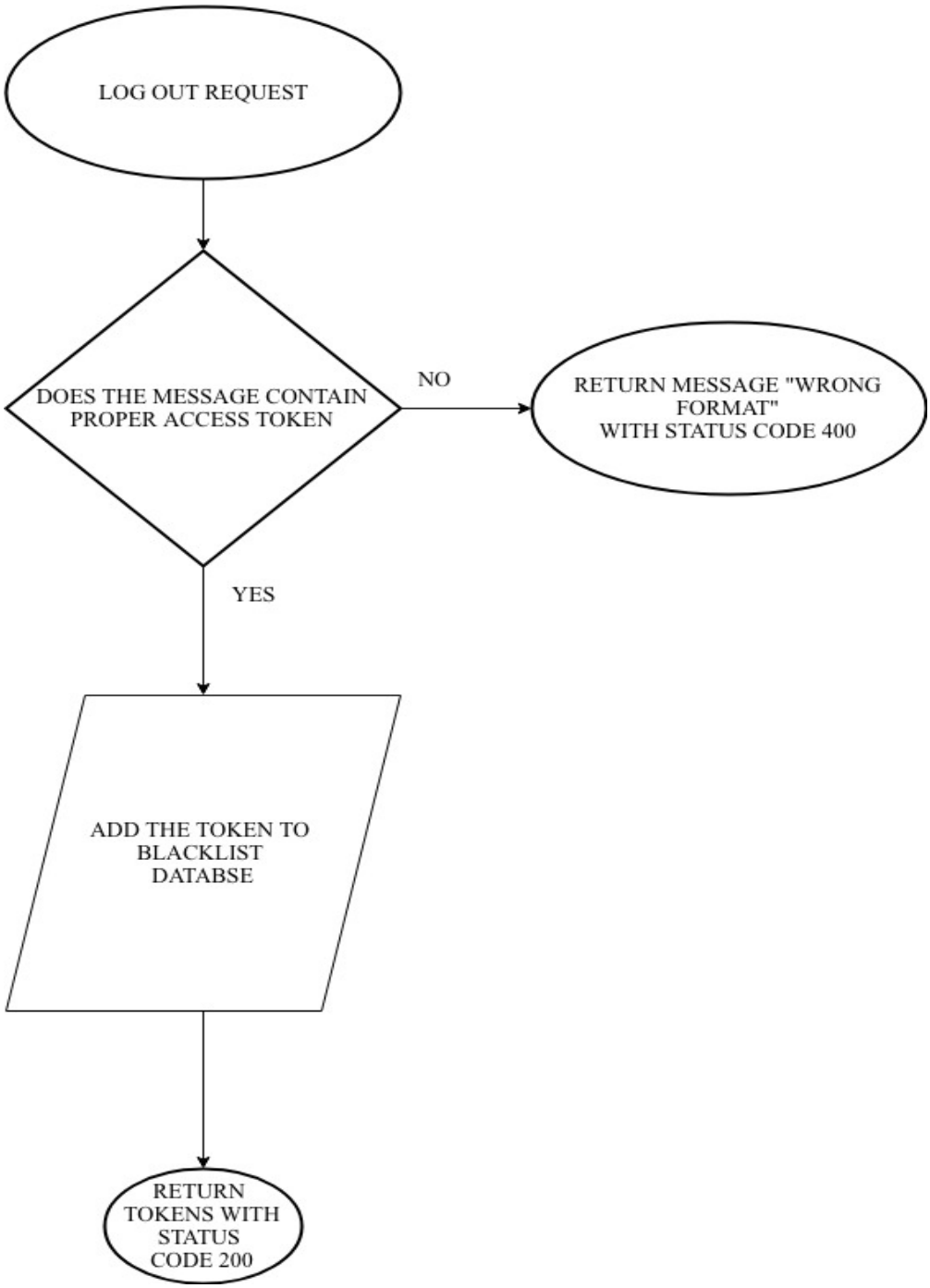

Fig. 2.6 Flow Chart for Logout Operation

d) Forgot Password:

This allows the users to reset their passwords in case they have forgotten it.

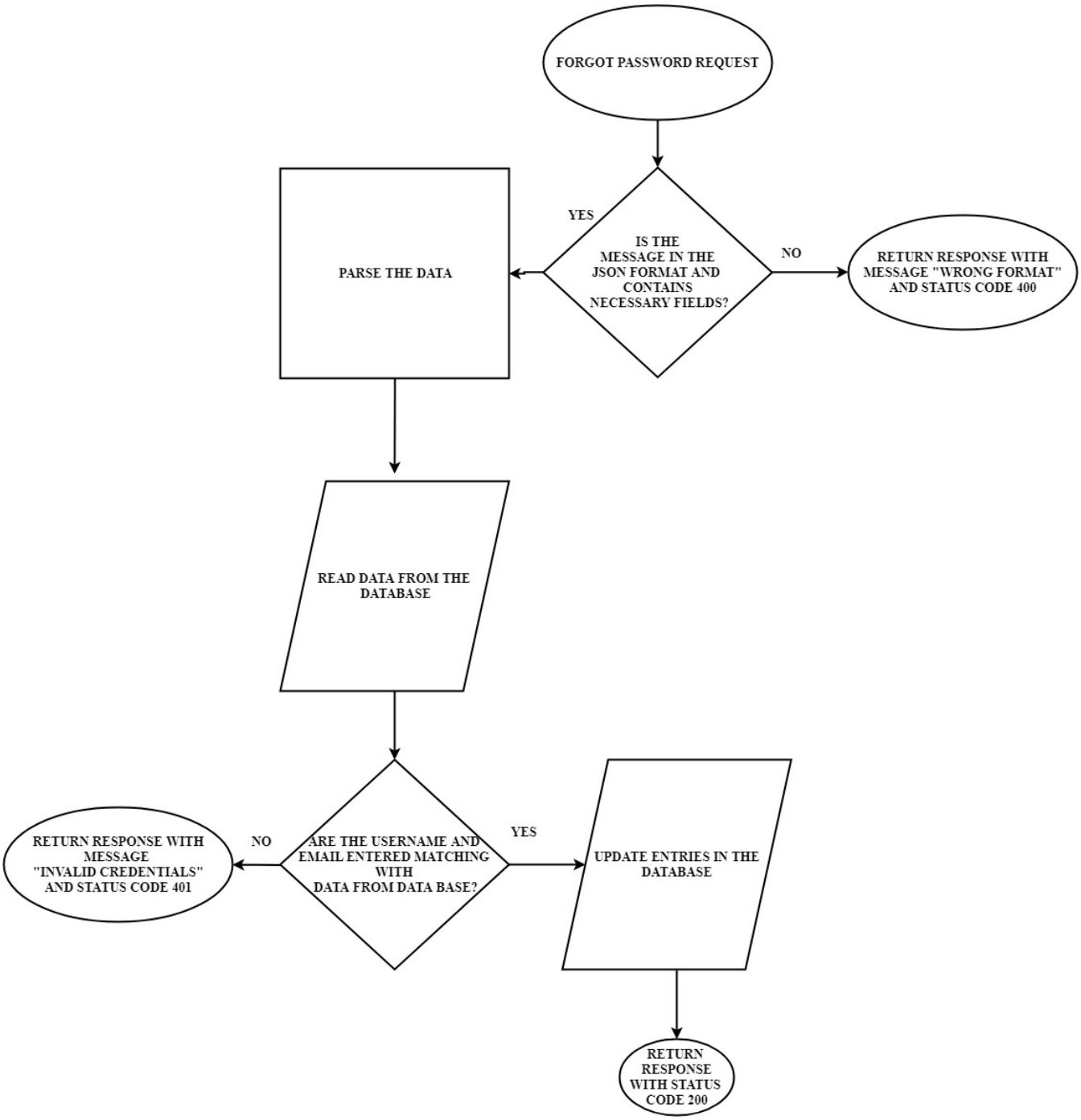

Fig. 2.7 Flow Chart for Forgot Password operation

e) Add Router:

This allows an Admin to add a router by providing the required credentials

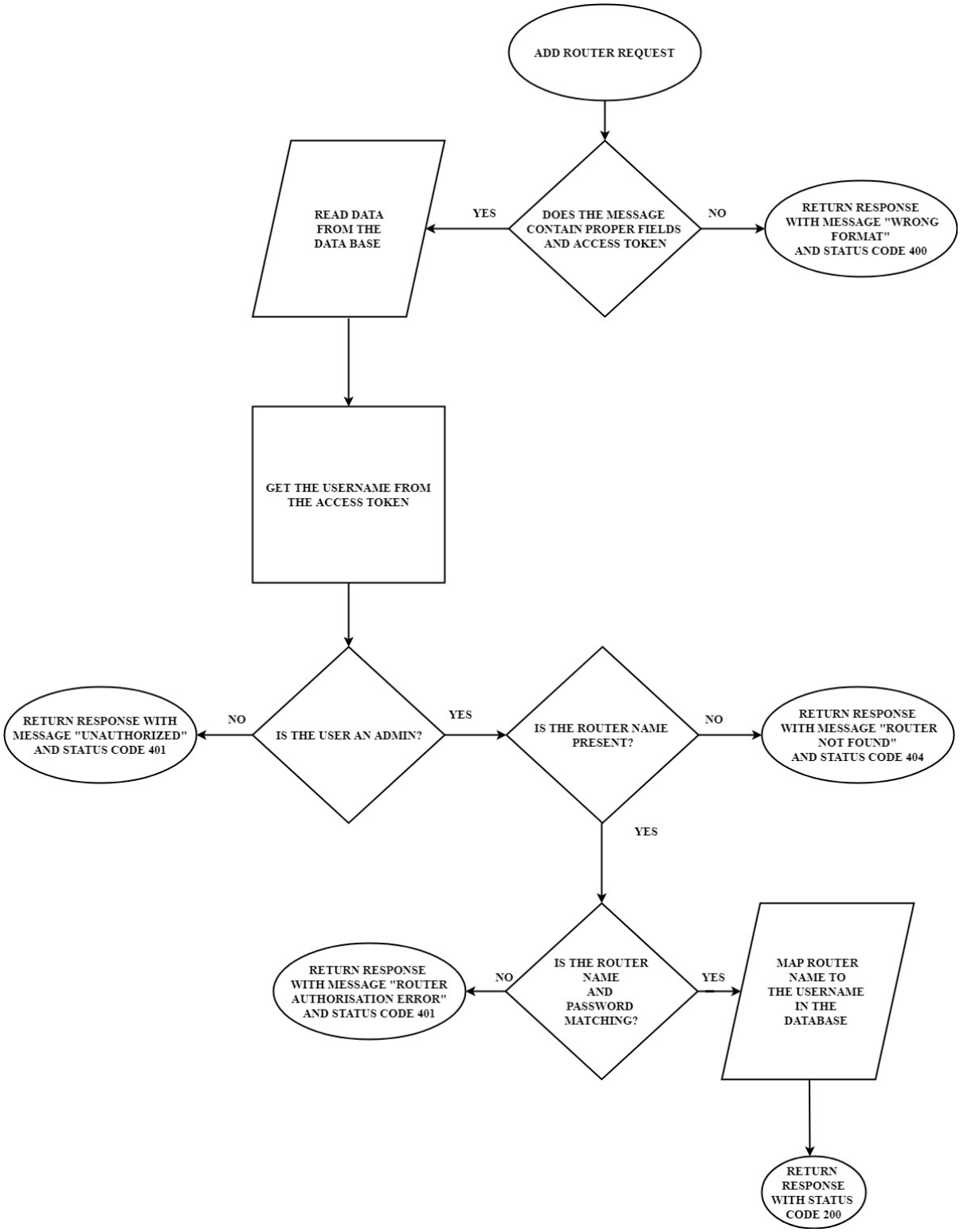

Fig. 2.8 Flow Chart for Add Router Operation

f) Remove Router:

This allows the user to remove the router control if necessary.

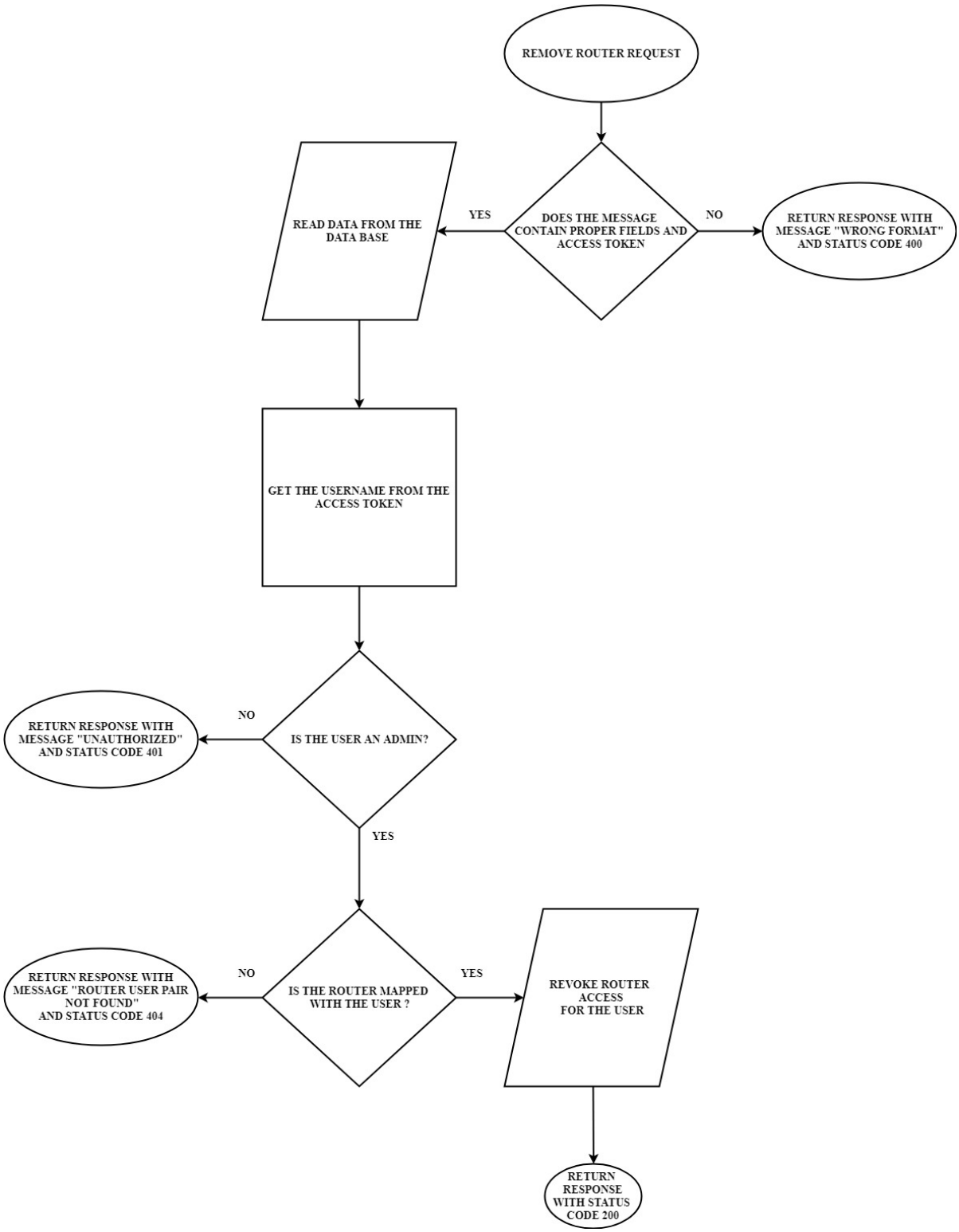

Fig. 2.9 Flow chart for Remove Router Operation

g) Add User:

This enables an Admin to add users under him.

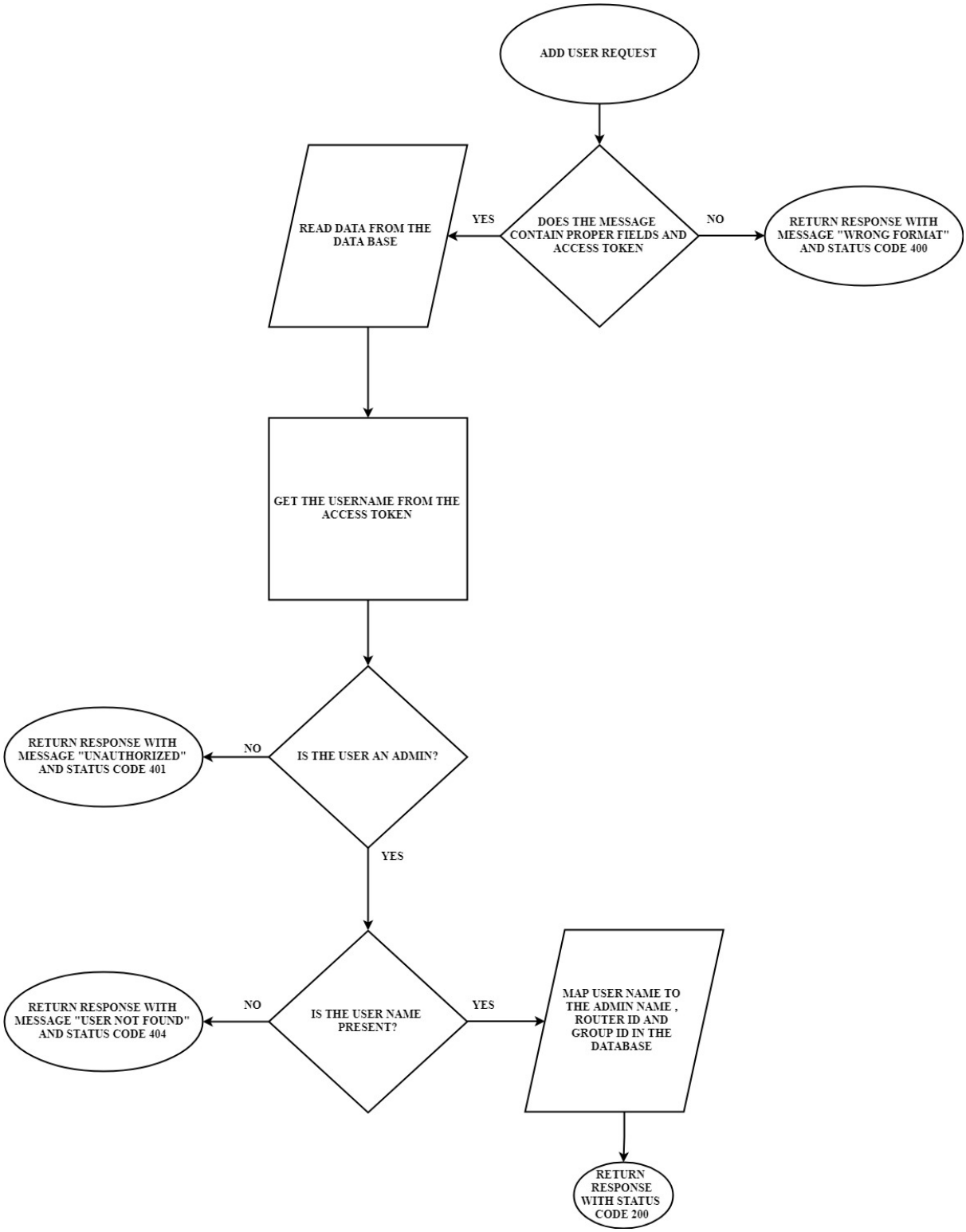

Fig. 2.10 Flow chart for Add User Operation

h) Remove User:

This allows an Admin to remove users under him.

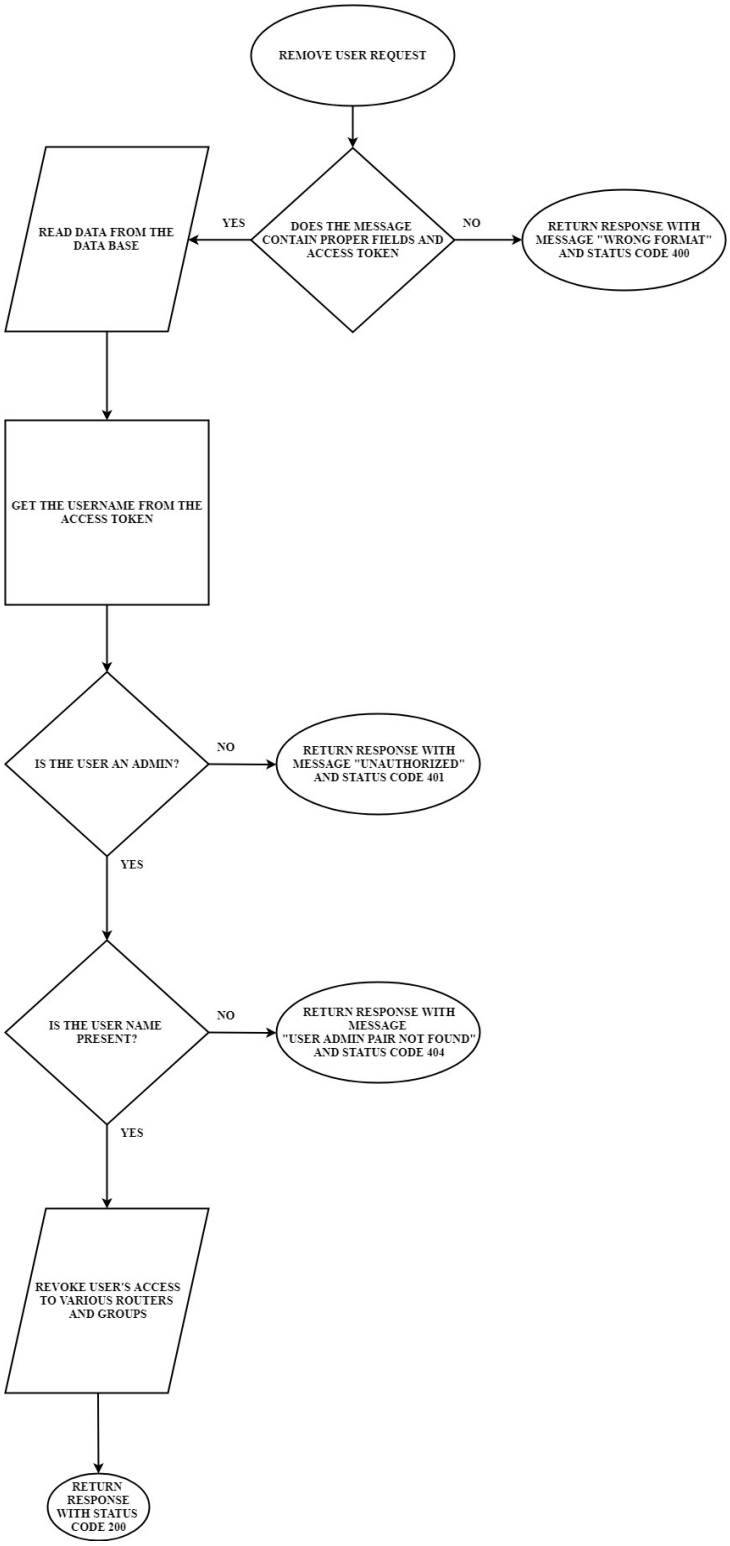

Fig. 2.11 Flow chart for Remove User Operation

i) Monitoring:

This allows the users to monitor the status of the devices that he/she is authorized for.

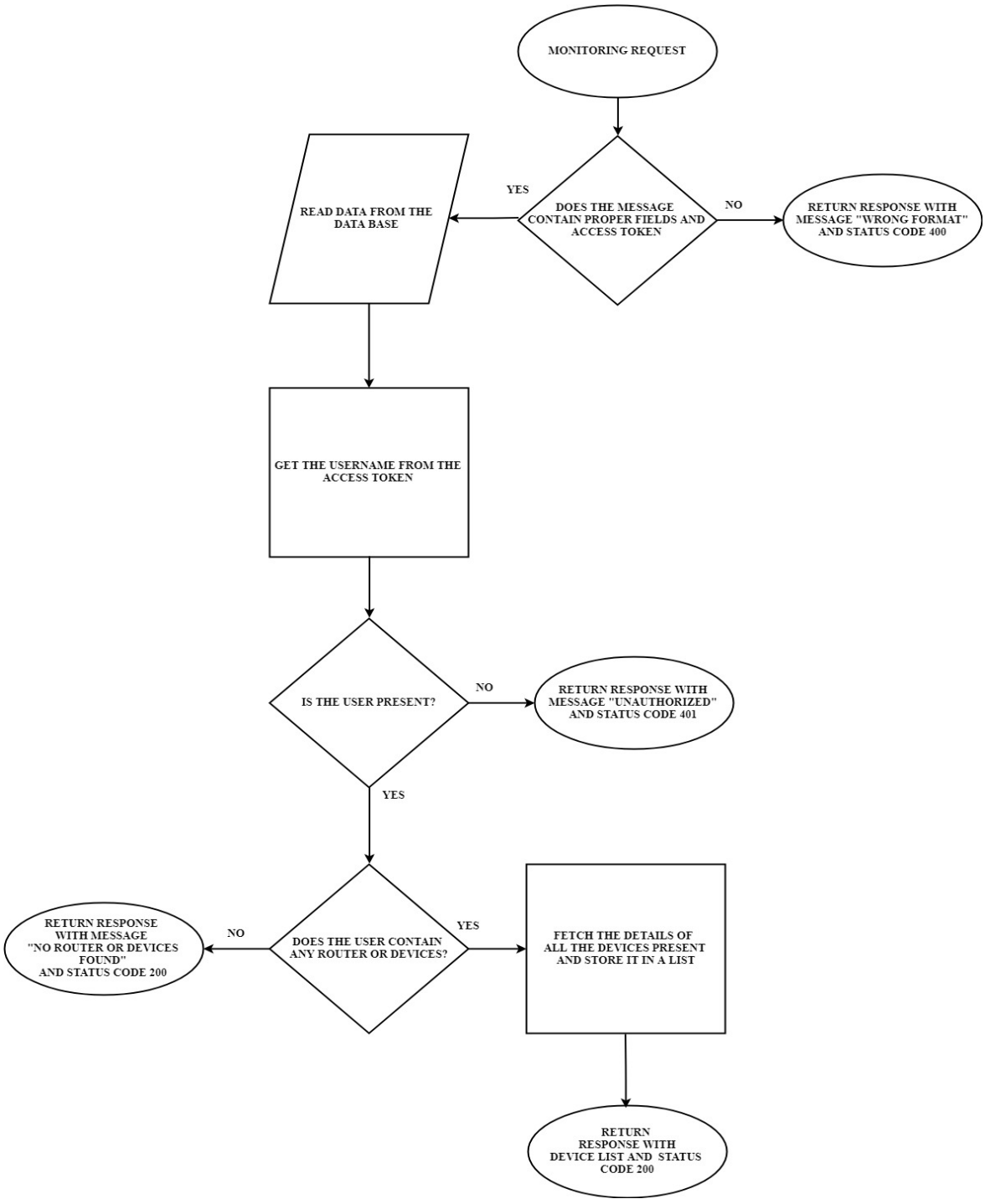

Fig. 2.12 Flow chart for Monitoring Operation

j) Controlling:

This allows users with credentials to control the devices they have been authorized for

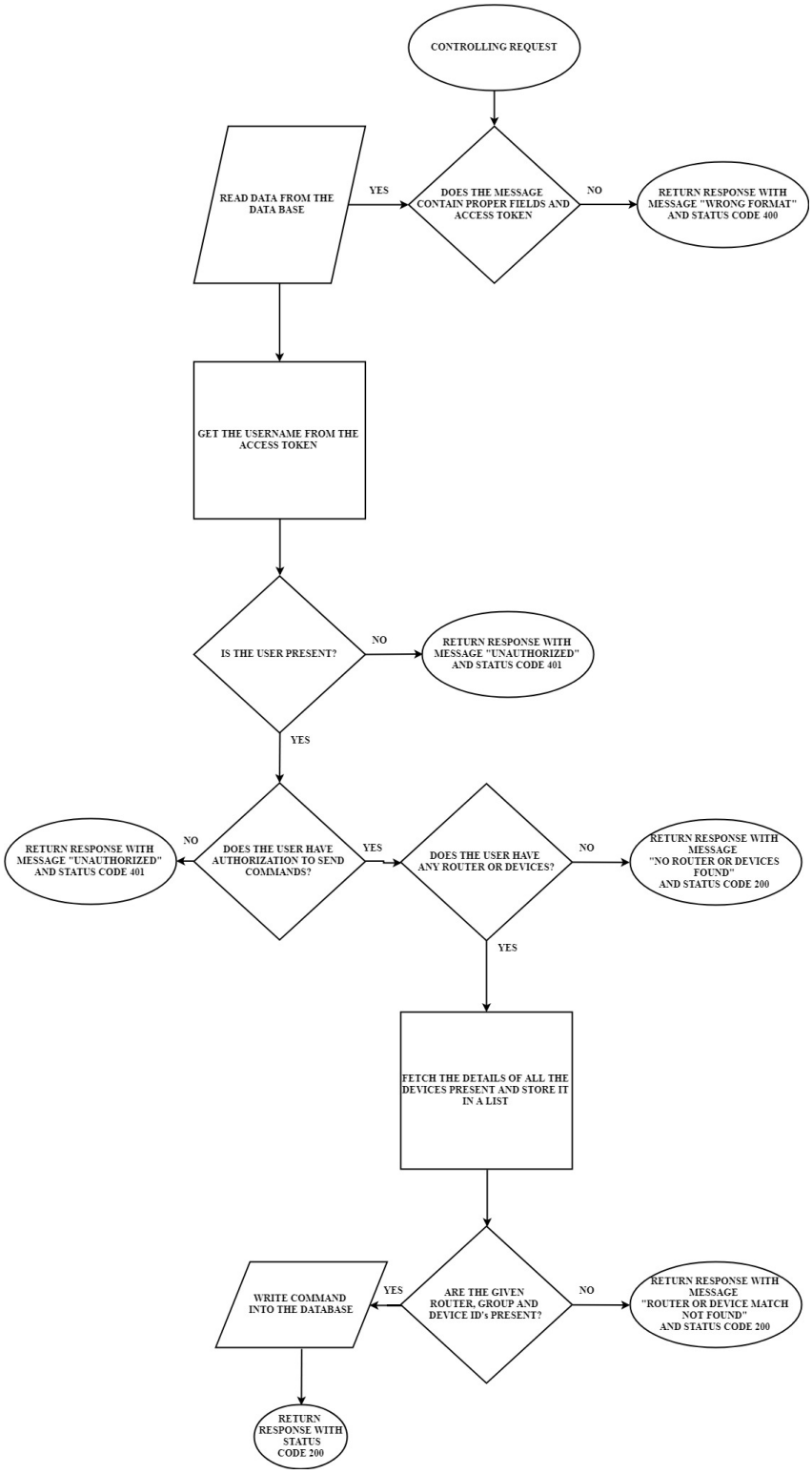

Fig. 2.13 Flow Chart Controlling Operation

2.3 MQTT:

2.3.1 INTRODUCTION TO MQTT:

MQTT (Message Queuing Telemetry Transport) is an ISO standard (ISO/IEC PRF 20922) publish-subscribe-based messaging protocol. It works on top of the TCP/IP protocol. It is designed for connections with remote locations where a "small code footprint" is required or the network bandwidth is limited. The publish-subscribe messaging pattern requires a message broker.

An MQTT system consists of clients communicating with a server, often called a "broker". A client may be either a publisher of information or a subscriber. Each client can connect to the broker. Information is organized in a hierarchy of topics. When a publisher has a new item of data to distribute, it sends a control message with the data to the connection broker. The broker then distributes the information to any clients that have subscribed to that topic. The publisher does not need to have any data on the number or locations of subscribers, and subscribers, in turn, do not have to be configured with any data about the publishers. If a broker receives a topic for which there are no current subscribers, it will discard the topic unless the publisher indicates that the topic is to be retained. This allows new subscribers to a topic to receive the most current value rather than waiting for the next update from a publisher. When a publishing client first connects to the broker, it can set up a default message to be sent to subscribers if the broker detects that the publishing client has unexpectedly disconnected from the broker. Clients only interact with a broker, but a system may contain several broker servers that exchange data based on their current subscribers' topics. A minimal MQTT control message can be as little as two bytes of data. A control message can carry nearly 256 megabytes of data if needed. There are fourteen defined message types used to connect and disconnect a client from a broker, to publish data, to acknowledge receipt of data, and to supervise the connection between

client and server. MQTT relies on the TCP protocol for data transmission. A variant, MQTT-SN, is used over other transports such as UDP or Bluetooth. MQTT sends connection credentials in plain text format and does not include any measures for security or authentication. This can be provided by the underlying TCP transport using measures to protect the integrity of transferred information from interception or duplication.

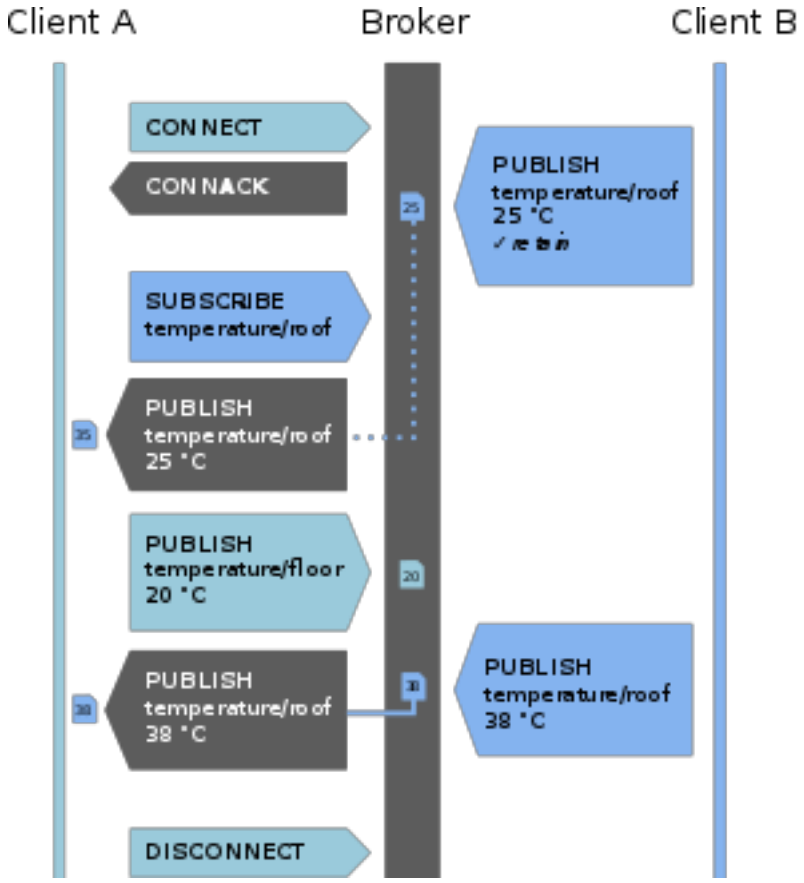

Fig. 2.14 Sample MQTT Broker and Clients

2.3.2 MQTT MESSAGE TYPES:

1. Connect:

   Waits for a connection to be established with the server and creates a link between the nodes

2. Disconnect:

   Waits for the MQTT client to finish any work it must do, and for the TCP/IP session to disconnect.

3. Publish:

   Publishes a message to a particular topic in a Broker.

4. Subscribe:

   Subscribes message from a particular topic from a Broker.

2.3.3 MQTT QUALITY OF SERVICE:

The Quality of Service (QoS) level is an agreement between the sender of a message and the receiver of a message that defines the guarantee of delivery for a specific message.

There are 3 QoS levels in MQTT:

1. QoS 0:

   At most once - the message is sent only once, and the client and broker take no additional steps to acknowledge delivery (fire and forget).

2. QoS 1:

   At least once - the message is re-tried by the sender multiple times until acknowledgment is received (acknowledged delivery).

3. QoS 2:

   Exactly once - the sender and receiver engage in a two-level handshake to ensure only one copy of the message is received (assured delivery.

2.3.4 MQTT BASIC DEFINITIONS:

a) Client:

An MQTT client is any device (from a microcontroller up to a full-fledged server) that runs an MQTT library and connects to an MQTT broker over a network.

b) Broker:

The counterpart of the MQTT client is the MQTT broker. The broker is at the heart of any publish/subscribe protocol. Depending on the implementation, a broker can handle up to thousands of concurrently connected MQTT clients. The broker is responsible for receiving all messages, filtering the messages, determining who is subscribed to each message, and sending the message to these subscribed clients.

2.3.5 MQTT SERVER SIDE CLIENT FLOW DIAGRAM:

The server-side client configuration for MQTT is shown in Fig. 2.15. The program connects to the broker and then performs required operations using callback functions.

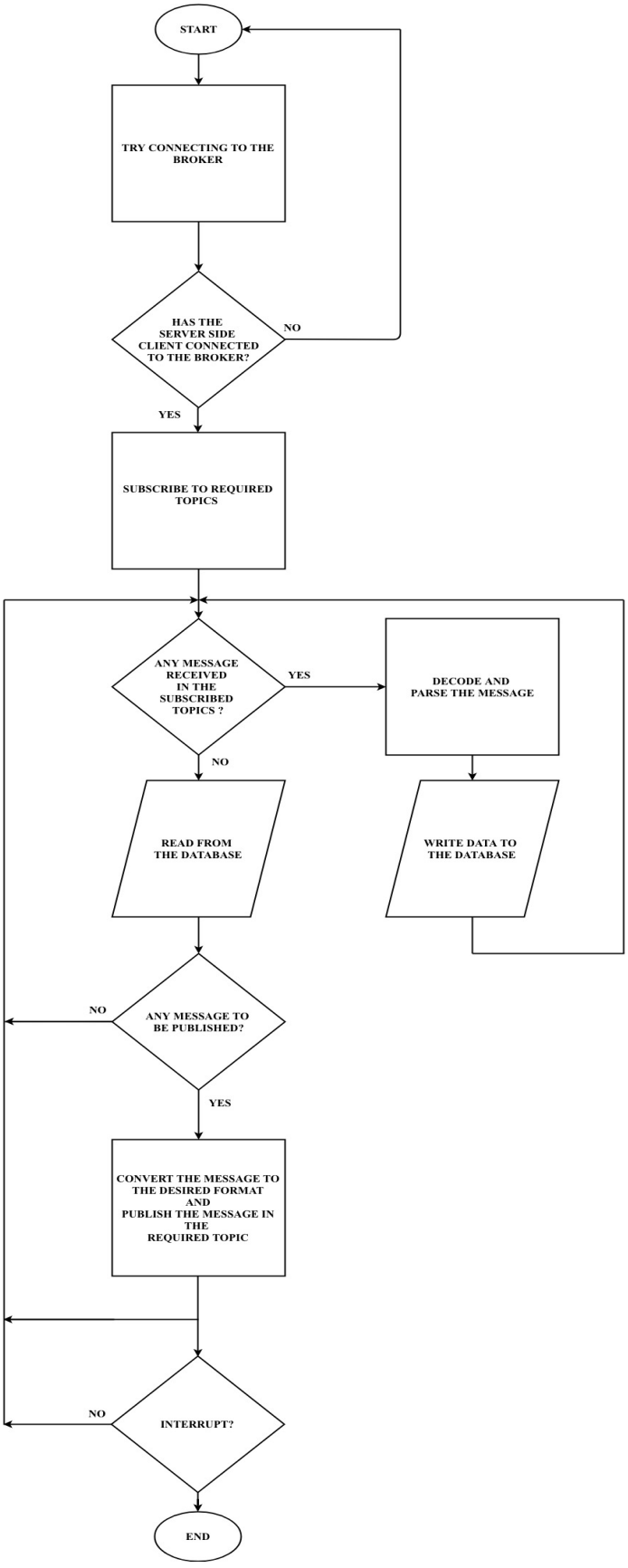

Fig. 2.15 Flow chart for MQTT Server Side Client

2.4 PRODUCTION ENVIRONMENT:

The entire backend is hosted on a production environment for the following reasons.

1. Handle multiple requests
2. Automatic restarting in case of server crash
3. Better response time

The production server used here is Nginx

Nginx is an Open Source Web Server that can be used as a reverse proxy, load balancer, mail proxy, and HTTP Cache.

Nginx maps external ports in a host to the local servers running in it. A Python Web Server Gateway Interface called as gunicorn is used to create clusters of the Flask app and interface it with Nginx.

Secure Socket Layer Certificates are used to encrypt the connections between the server and the app.

2.5 AWS IOT:

AWS IoT is a platform that enables you to connect devices to AWS Services and other devices, secure data and interactions, process and act upon device data and enable applications to interact with devices even when they are offline.

It provides the following features and products:

- AWS IOT Device SDK
- Device Gateway
- Authentication and Authorization
- Registry

- Device Shadows
- Rules Engine

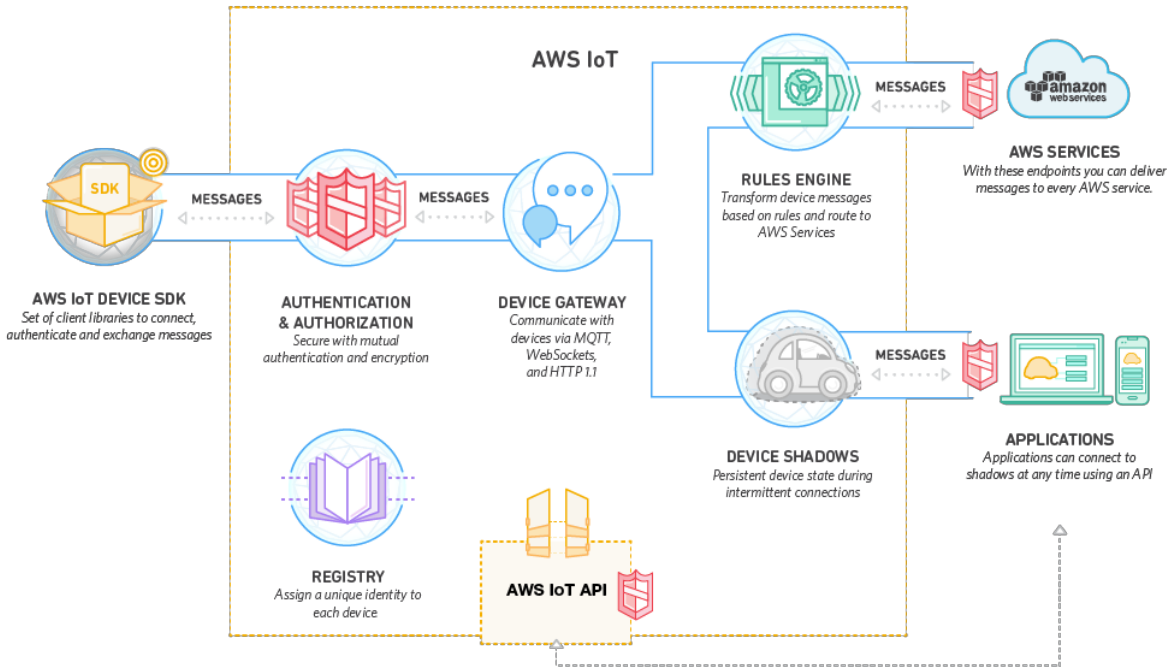

Fig. 2.16 Amazon Web Services IOT Features

Courtesy: Amazon Web Services

2.6 API TESTING:

API Testing is a very important part for successful implementation and to prevent errors after integrating into the app.

a) Sign Up:

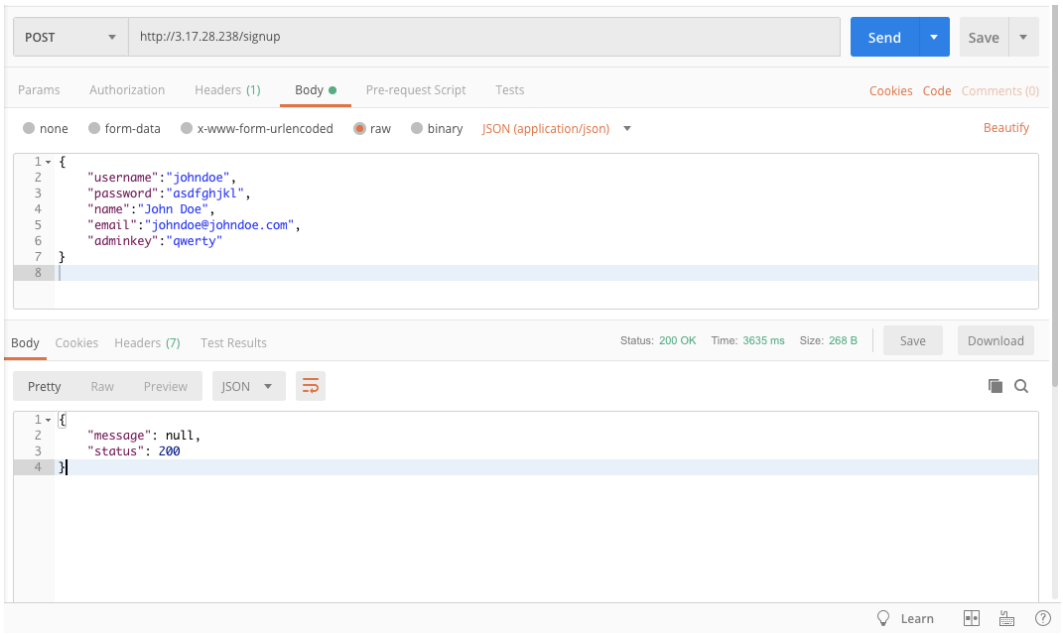

Fig. 2.17 Successful Sign Up Request and Response for an Admin

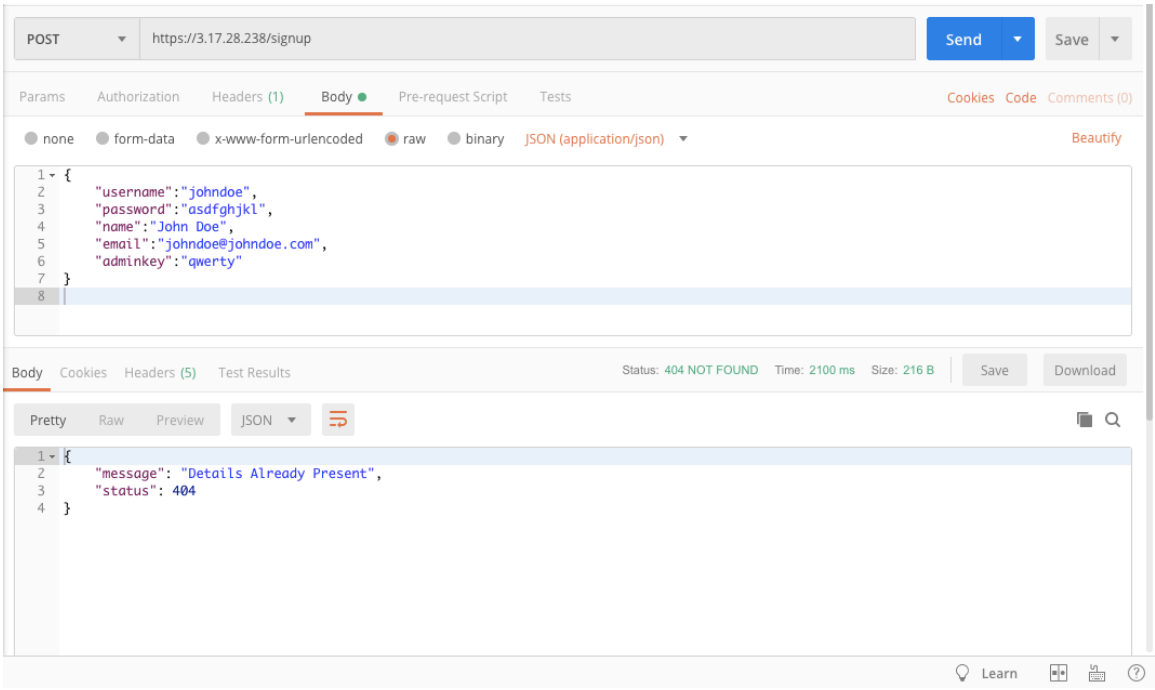

Fig. 2.18 Sign Up Request and Response when similar user is already present

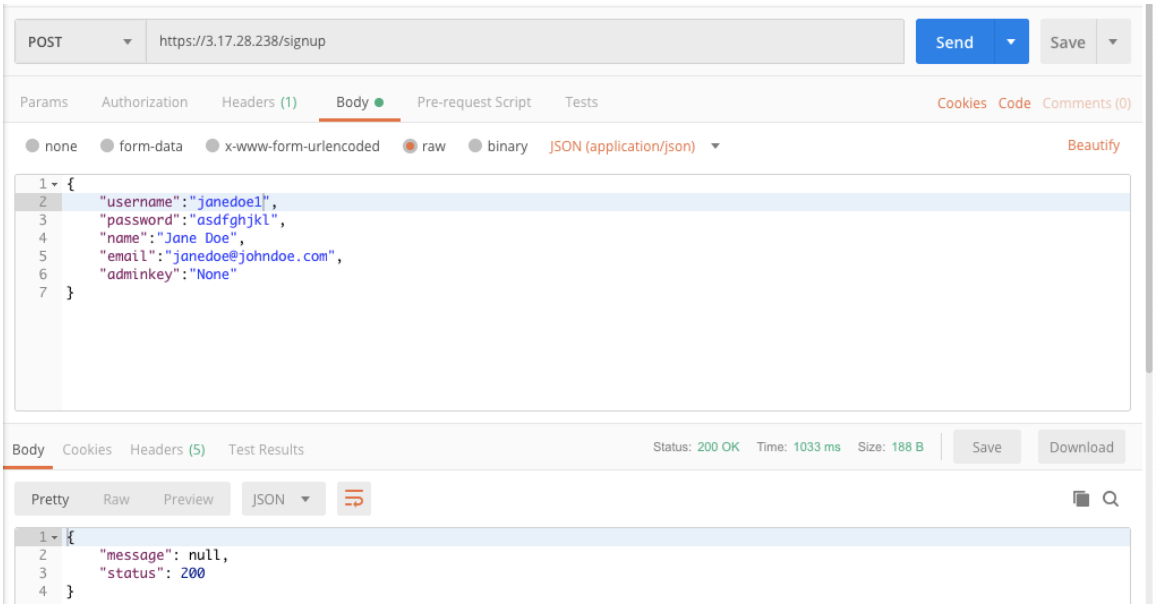

Fig. 2.19 Sign Up Request and Response for User

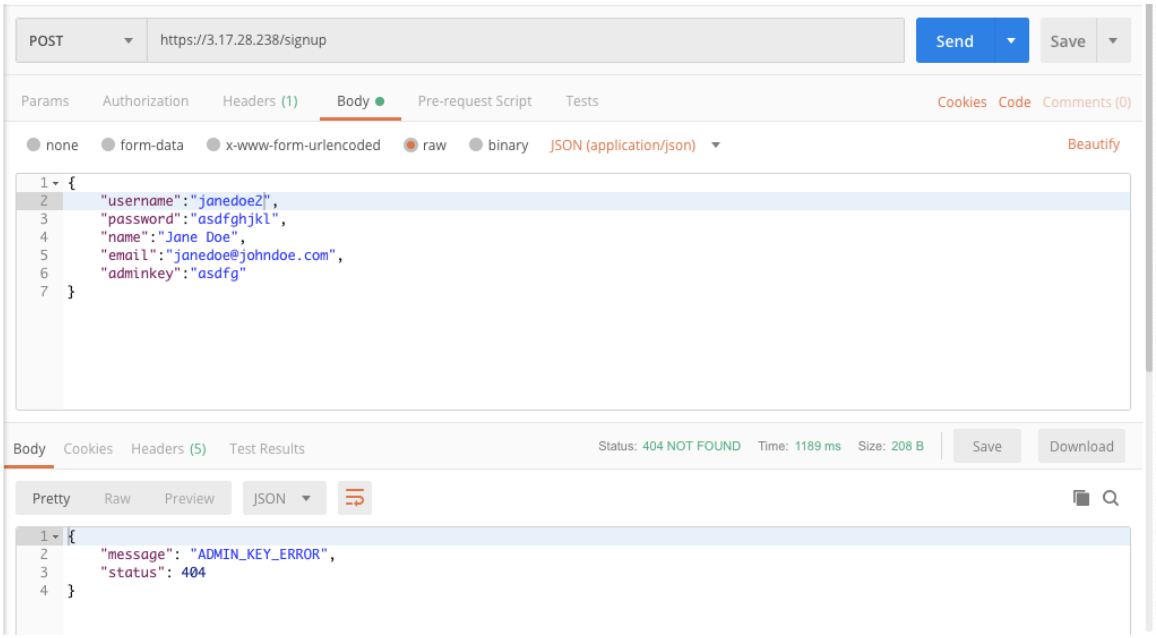

Fig. 2.20 Sign Up Request and Response when Admin Key becomes an error

The above diagrams give the detailed Request and Response for Sign Up Request.

b) Login:

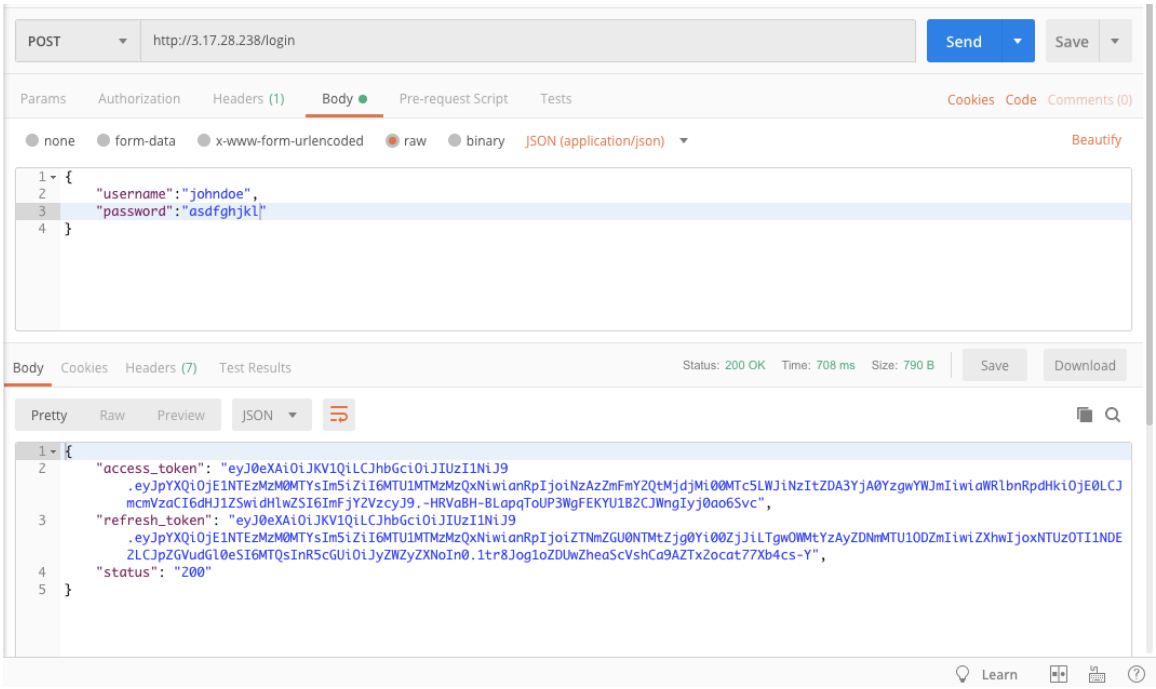

Fig. 2.21 Request and Response for a successful login

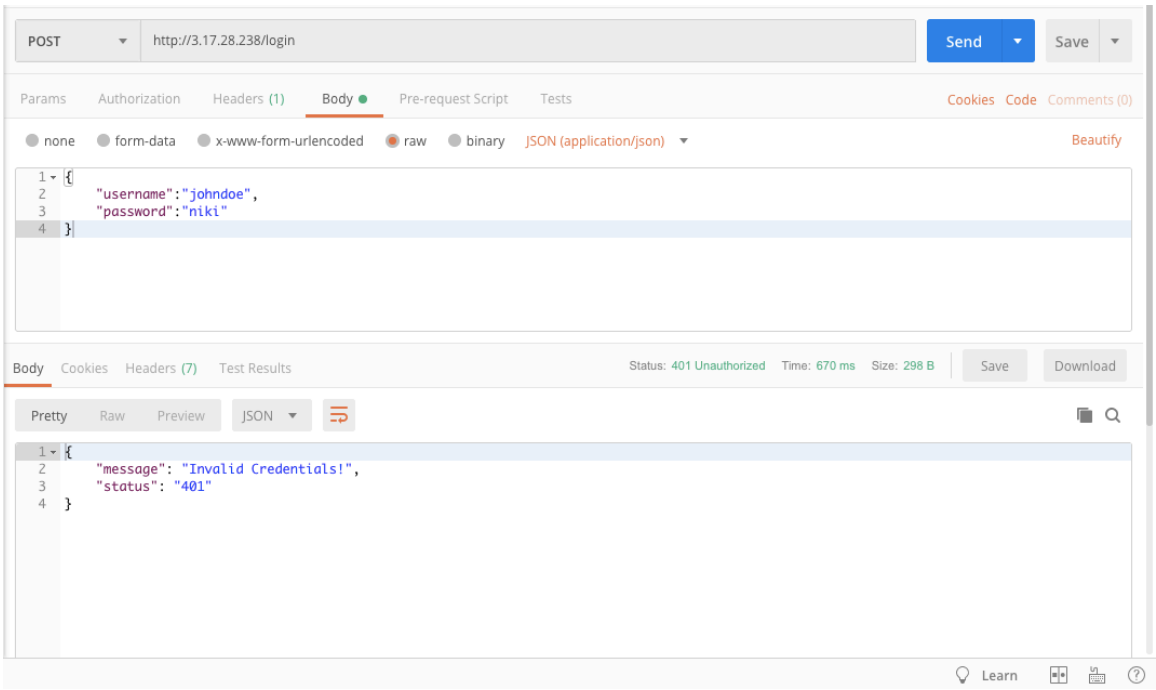

Fig. 2.22 Log in with Invalid Credentials

The above diagrams give the detailed Request and Response for Log In.

c) Log Out:

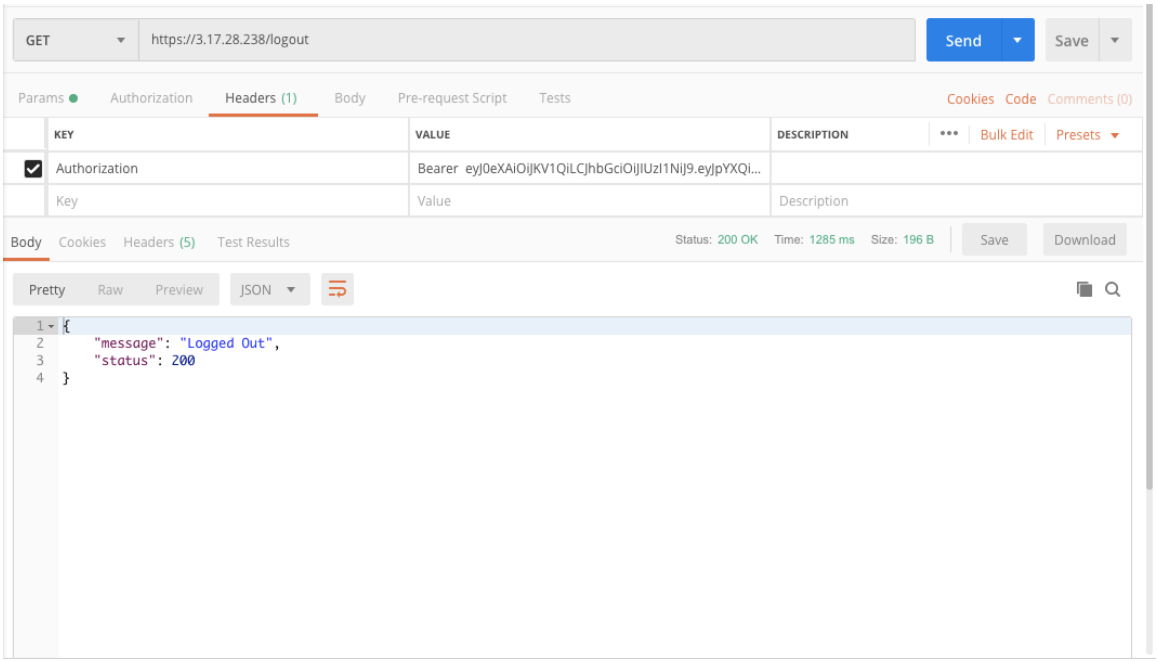

Fig. 2.23 Request and Response for Log Out

The above diagram gives the detailed Request and Response for Log Out

d) Forgot Password:

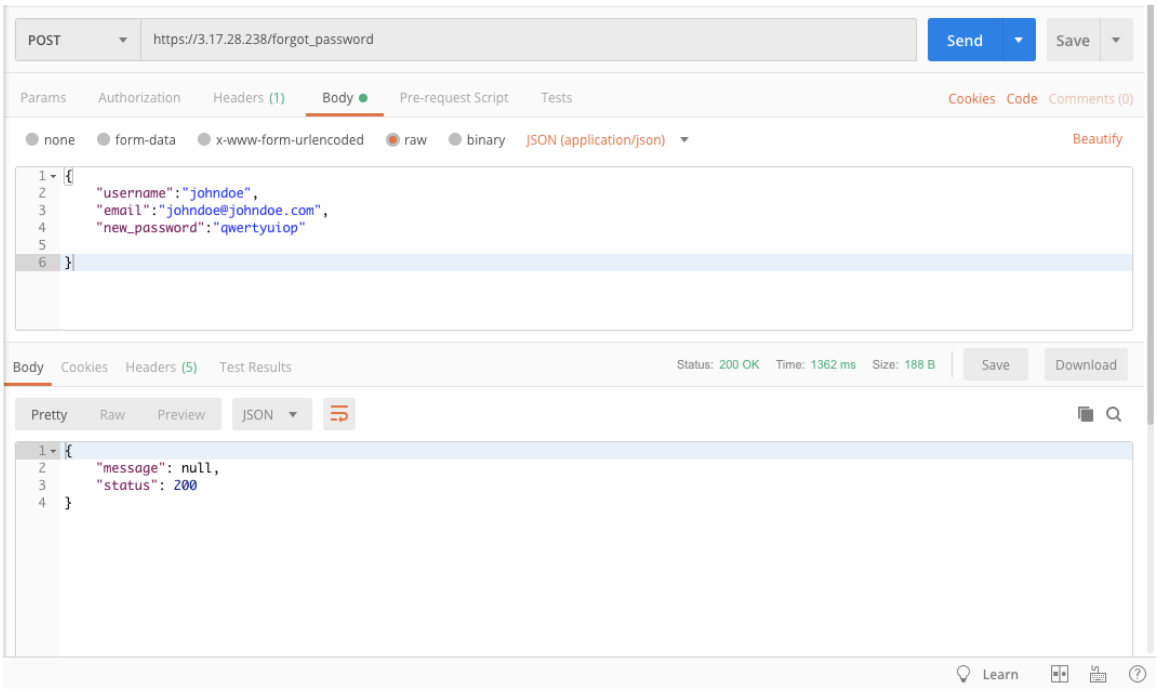

Fig. 2.24 Forgot Password Request and Response

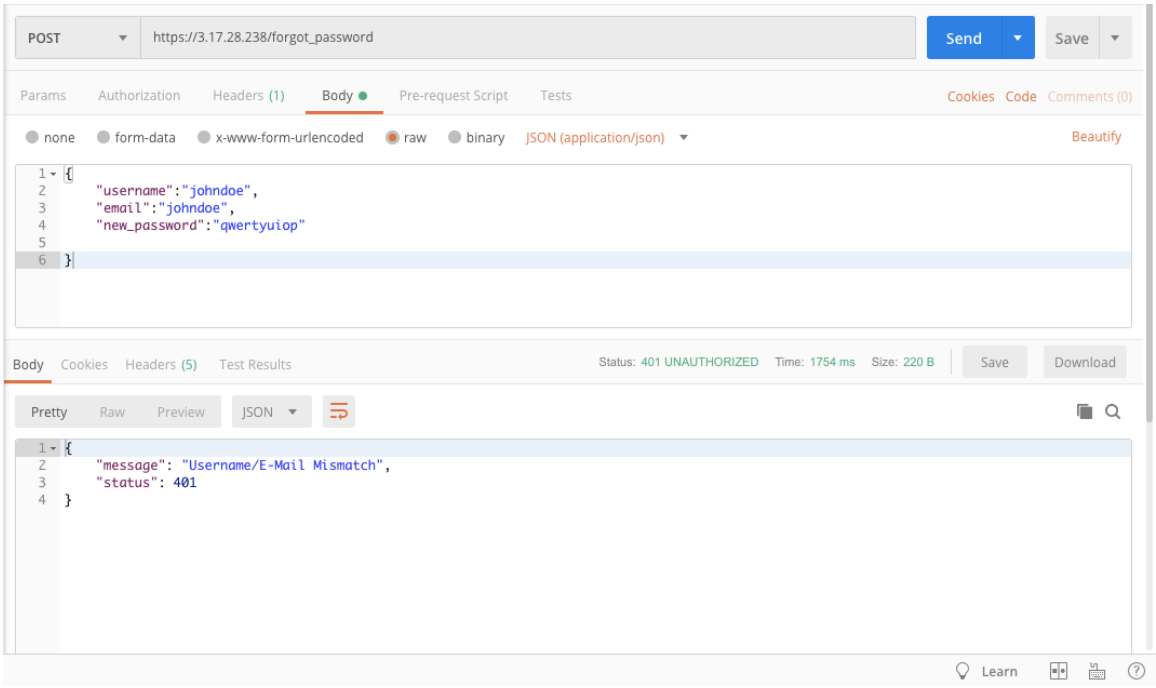

Fig. 2.25 Forgot Password Request and Response with Invalid Credentials

The above diagram gives a detailed Request and Response for Forgot Password

e) Add Router:

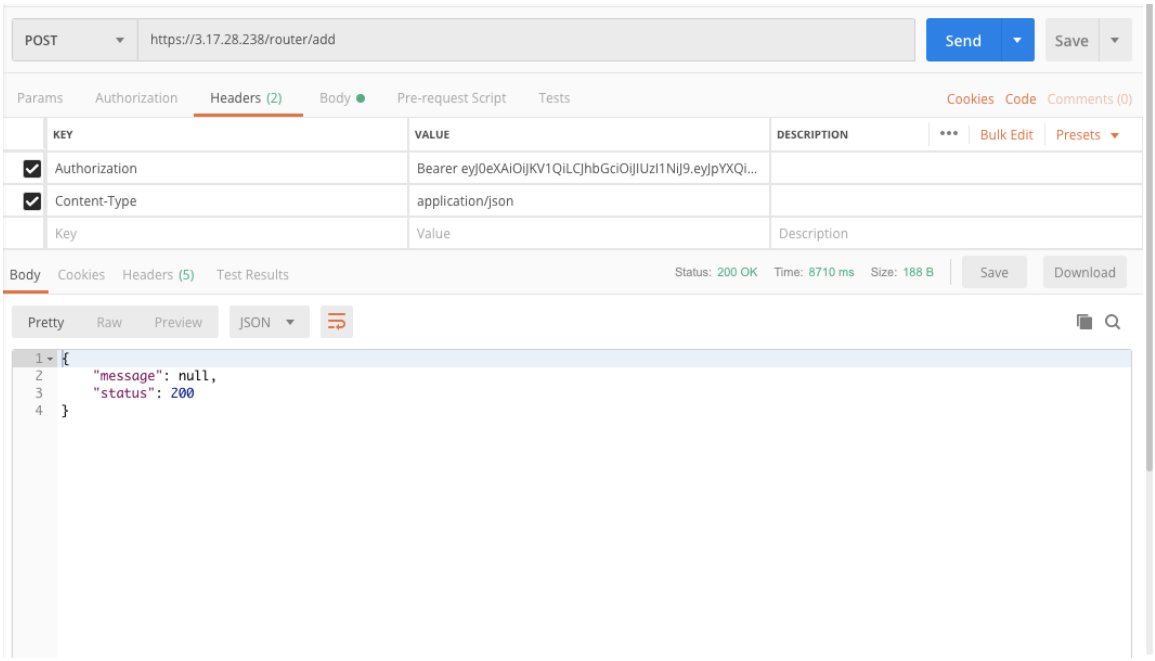

Fig. 2.26 Add Router Request and Response successful

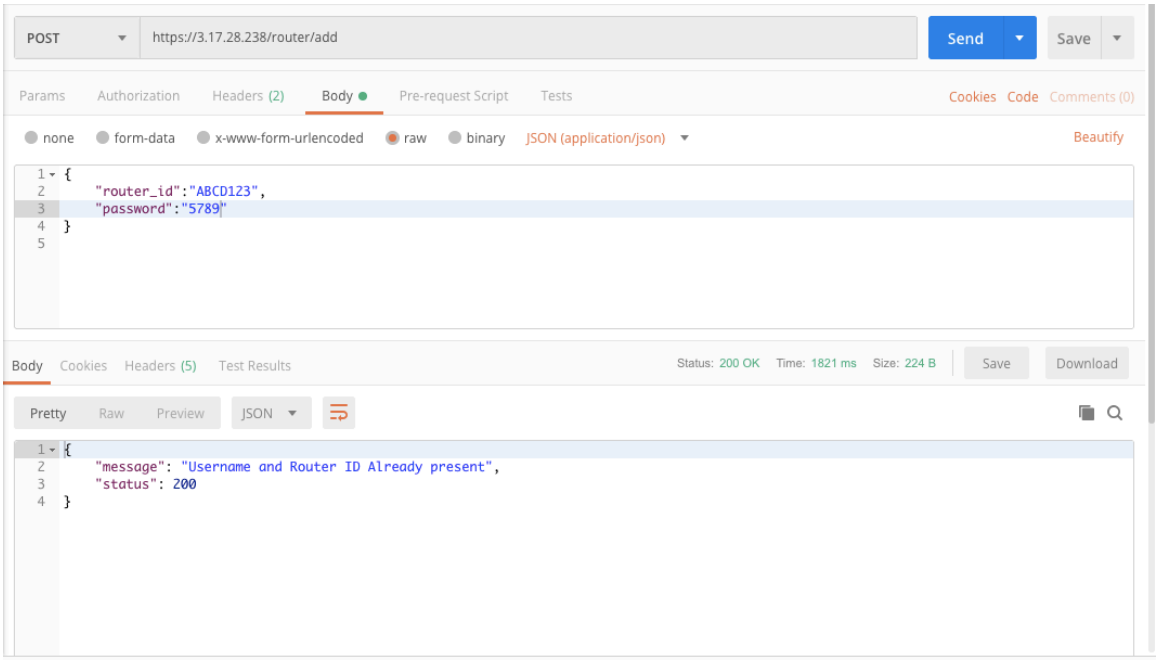

Fig. 2.27 Add Router Request and Response when adding again

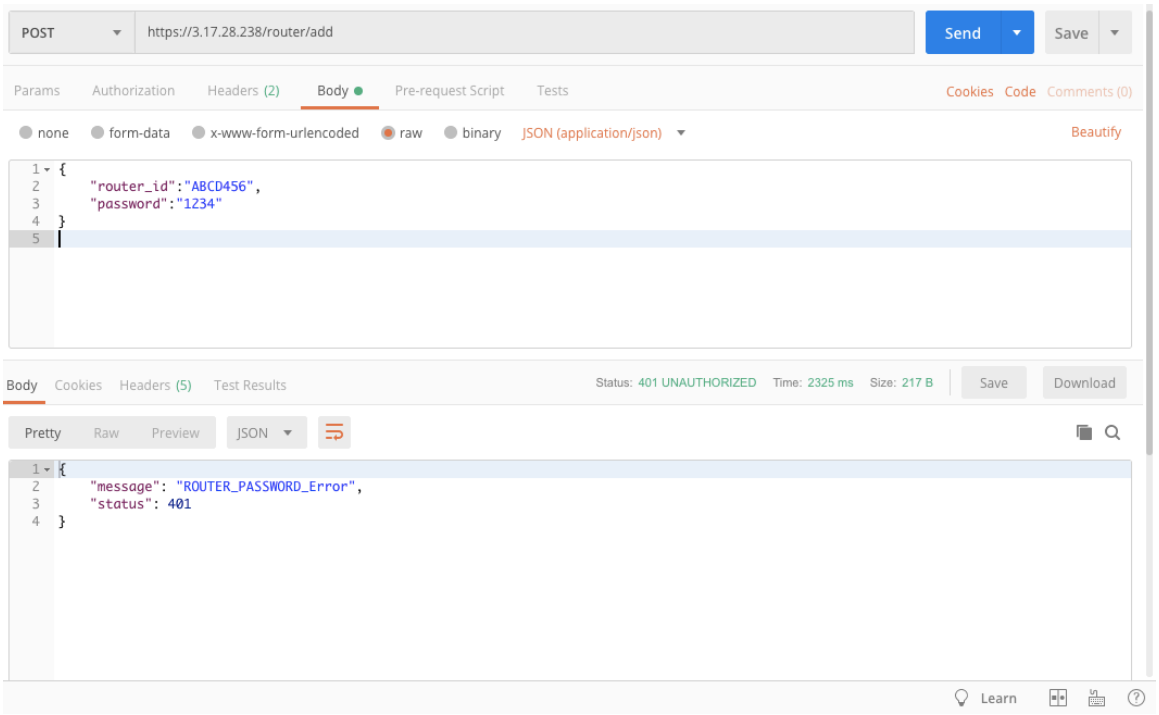

Fig. 2.28 Add Router Request and Response with wrong Router Password

The above diagram gives a detailed Request and Response for Add Router.

f) Remove Router:

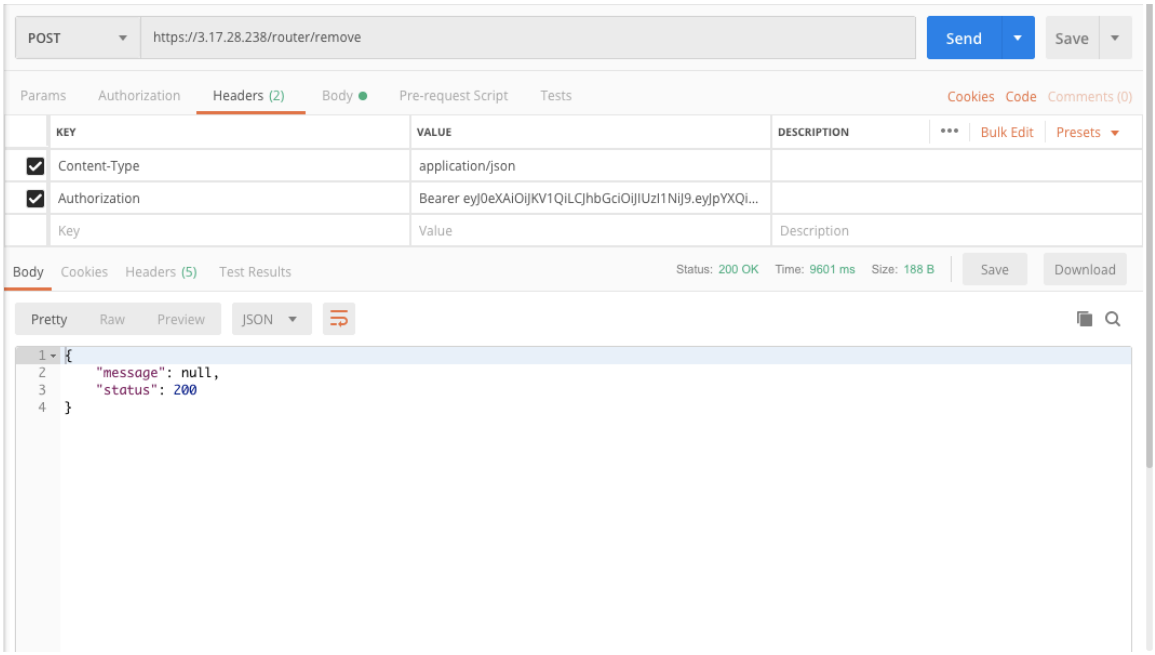

Fig. 2.29 Remove Router Request and Response

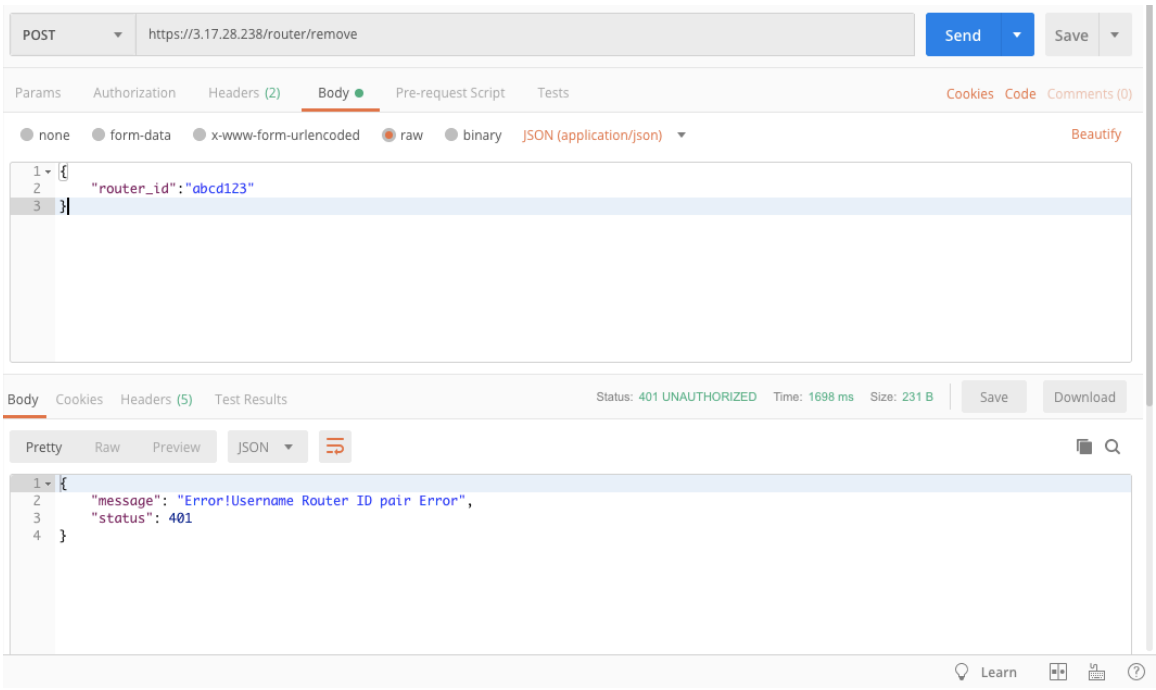

Fig. 2.30 Remove Router Request and Response with wrong Router

The above diagram gives a detailed Request and Response for Remove Router

g) Add User:

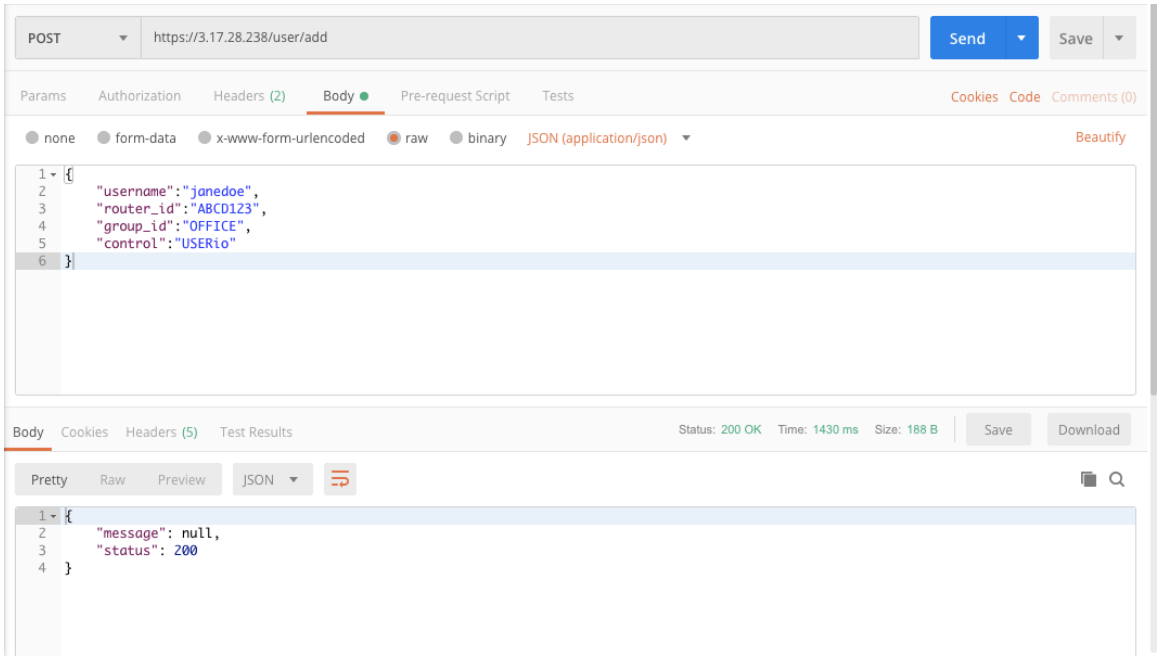

Fig. 2.31 Add User Request and Response

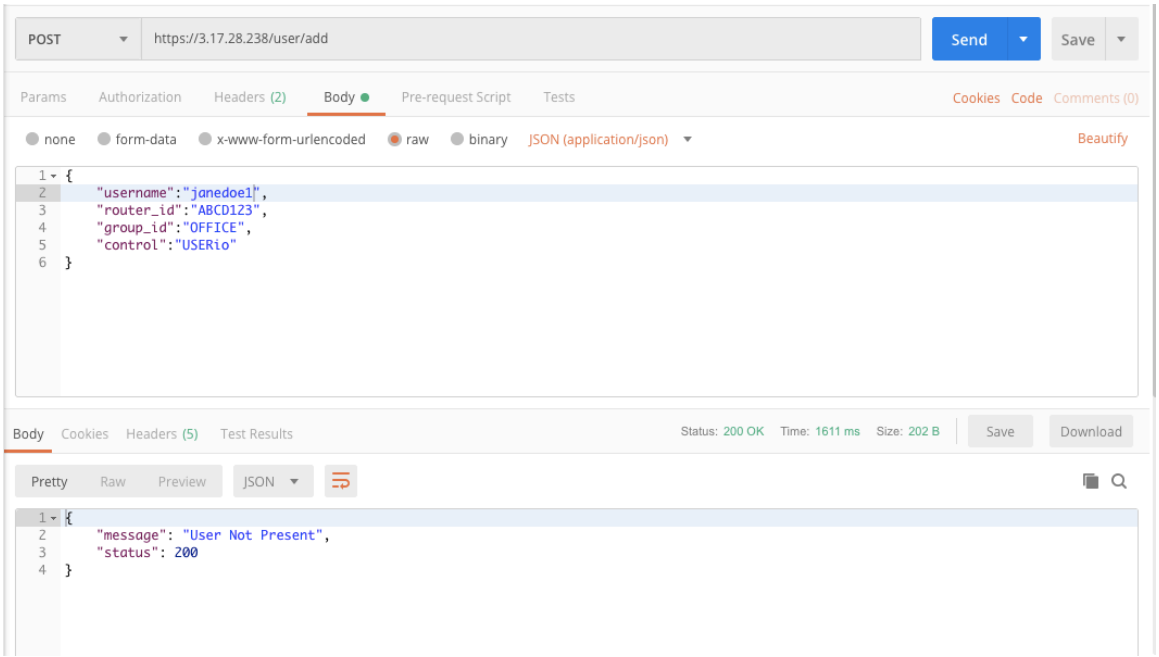

Fig. 2.32 Add User Request and Response when the user details are not present

The above diagrams give the detailed Request and Response for Add User

h) Remove User:

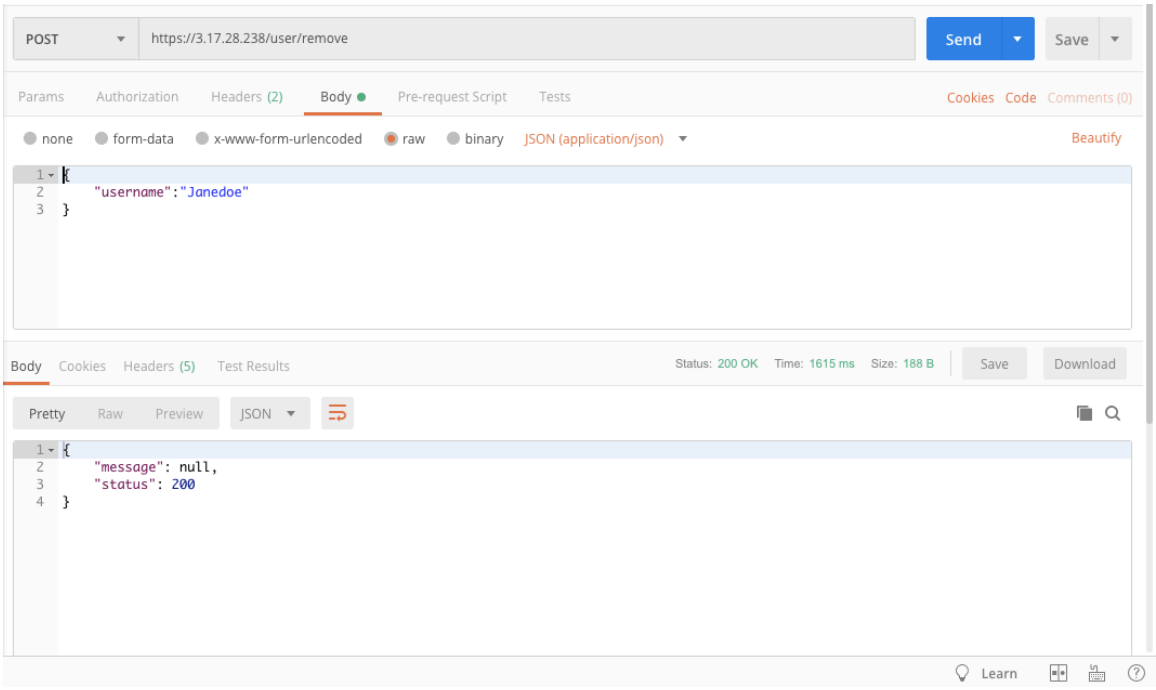

Fig. 2.33 Remove User Request and Response

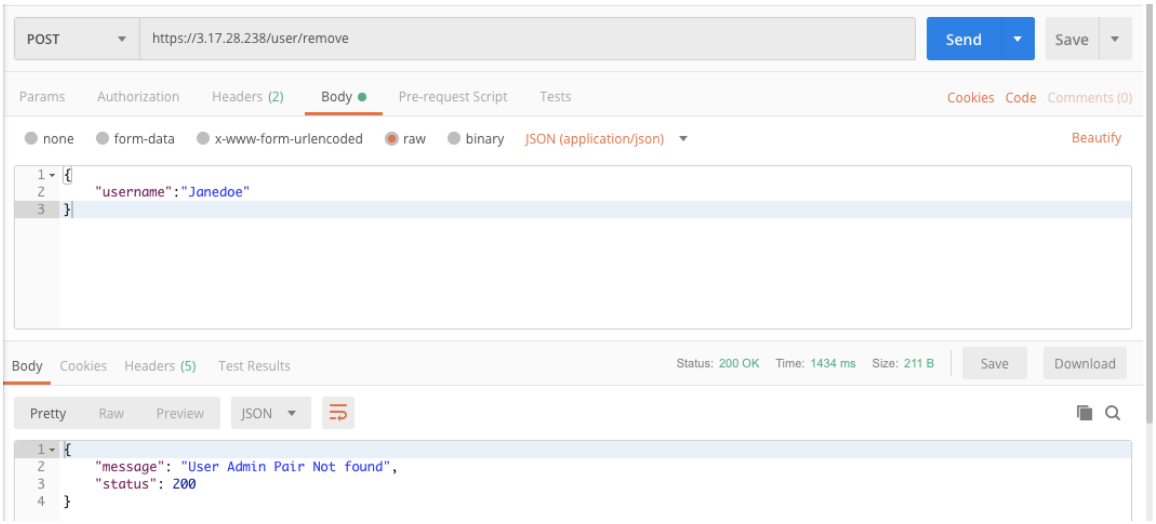

Fig. 2.34 Remove User Request and Response when the username is not present

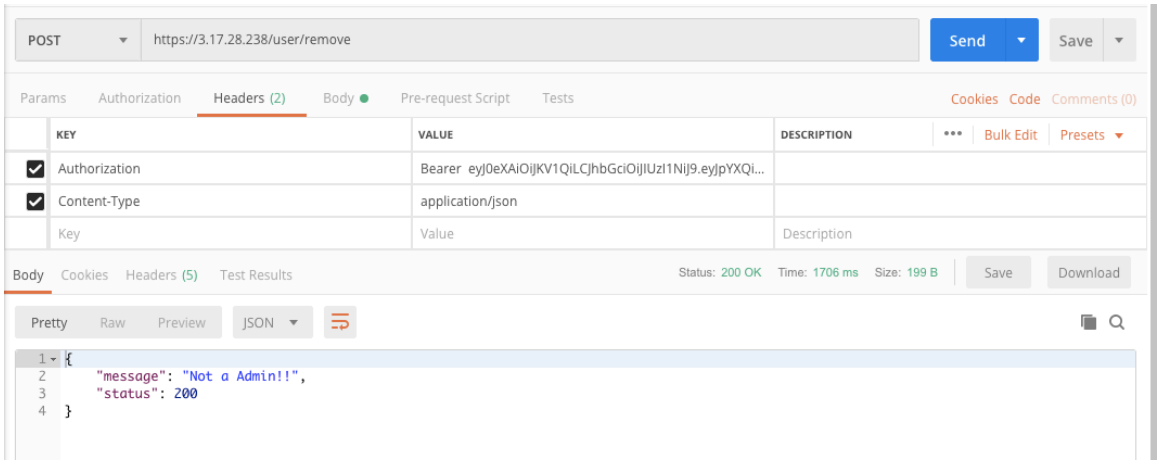

Fig. 2.35 Remove User Request and Response when not a user does it

The above diagrams give the detailed Request and Response for Remove User.

i) Monitoring:

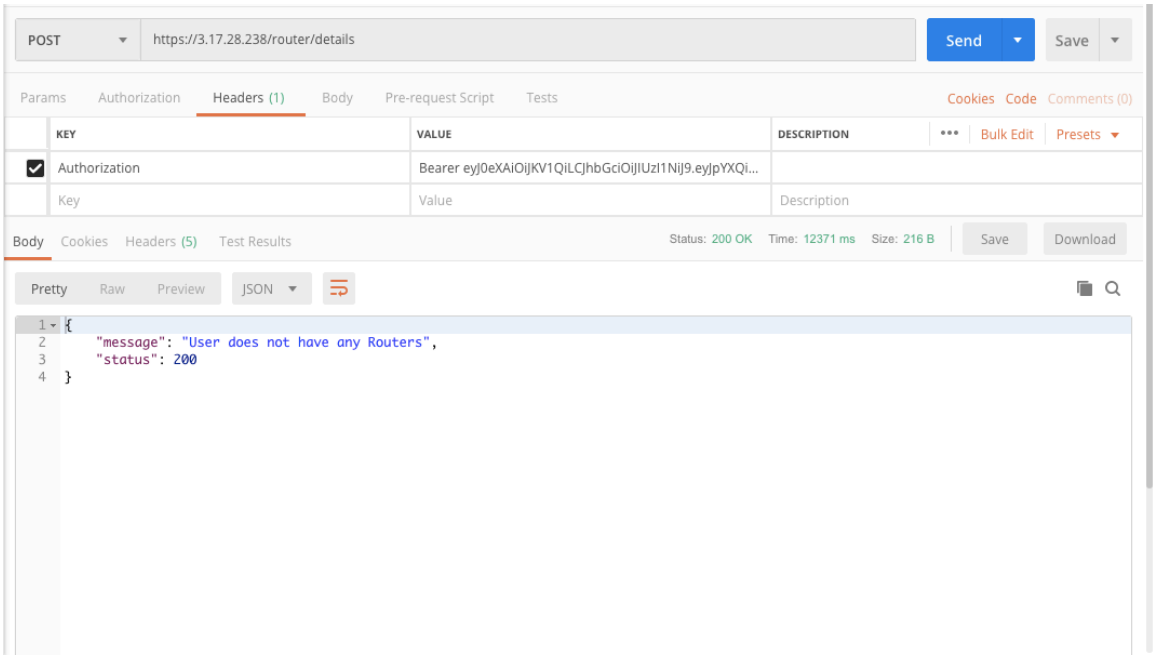

Fig. 2.36 Monitoring Request and Response when user does not have any Routers

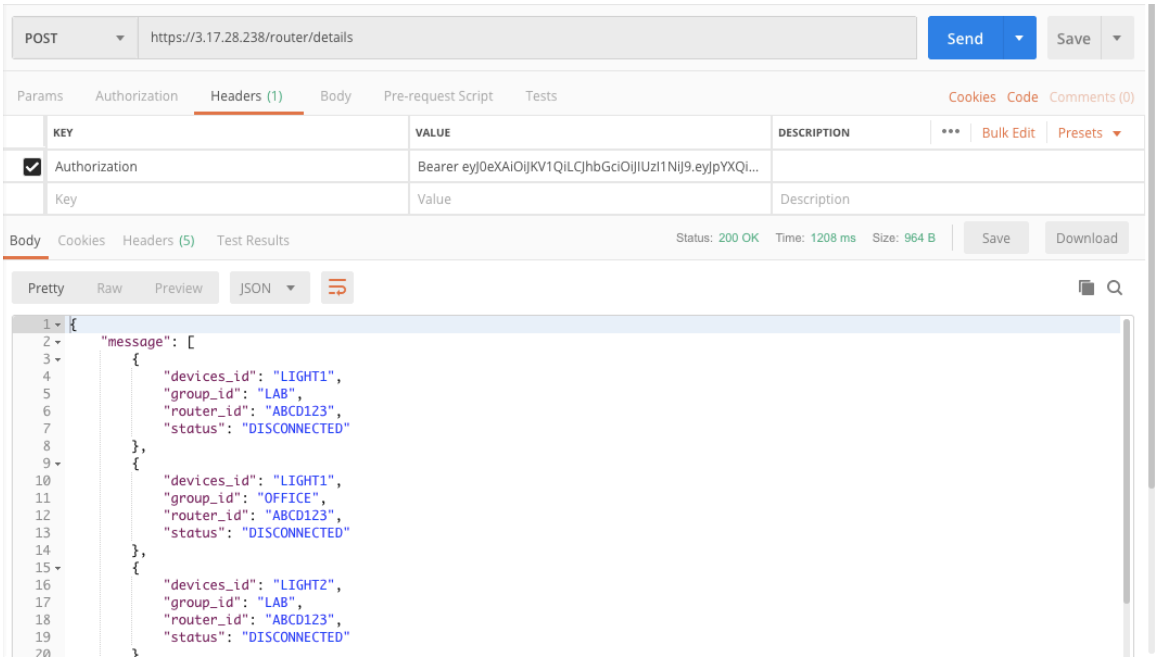

Fig. 2.37 Monitoring Request and Response when user has devices

The above diagram gives a clear idea about what is the format for receiving the details of the devices

j) Controlling:

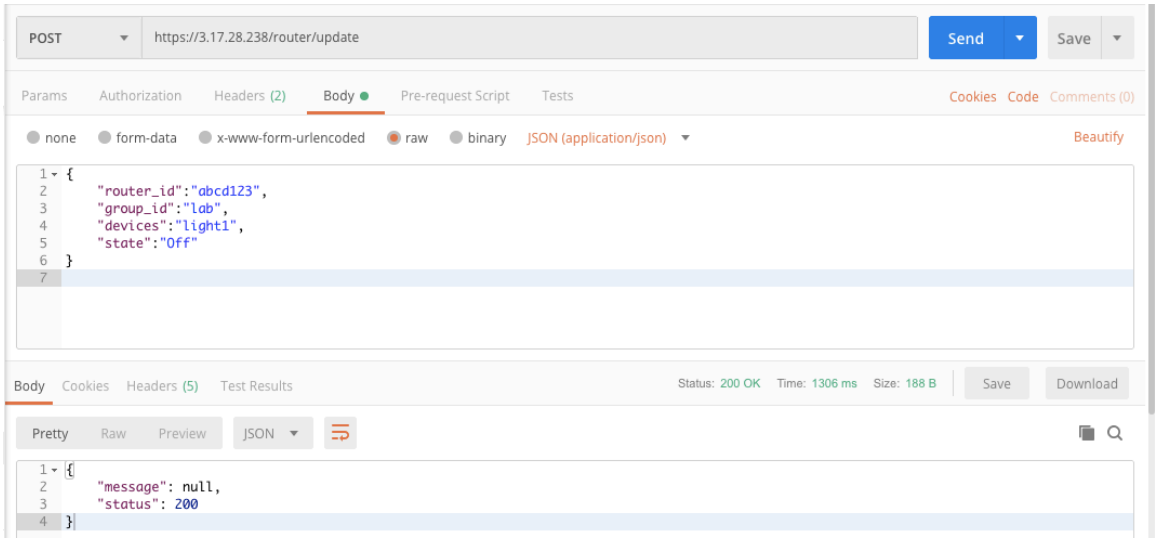

Fig. 2.38 Controlling Request and Response

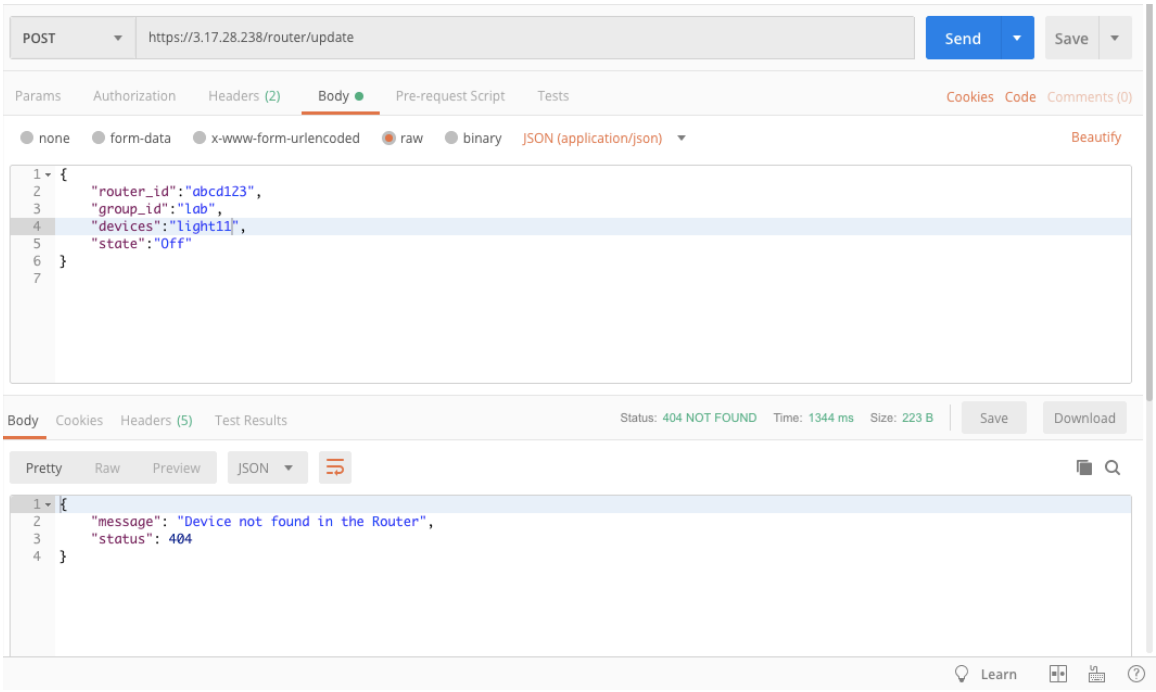

Fig. 2.39 Controlling Request and Response with wrong Device ID

The above diagrams give the detailed Request and Response for sending commands to devices

CONCLUSION:

A backend has been designed, implemented and tested successfully for performing the various operations of Sign Up, Log In, Log Out, Add Router, Remove Router, Monitoring, Controlling.

CHAPTER 3

USER INTERFACE FOR ANDROID OS

3.1 INTRODUCTION:

An android app named ThingSTalk was created using android studio IDE. The app basically has the following layouts and widgets which enables the system to support the proposed functionalities.

3.2 ACTIVITY 1:

The first page is basically a splash screen which serves a welcome page for the user interface. A normal constraint layout has been used for this activity.

Fig. 3.40 Welcome Page

3.3 ACTIVITY 2:

The second activity prompts the user to either signup- creating a new account with the app by giving valid credentials or sign in- enter into an already existing account by typing only the required fields. A forgot password option has also been

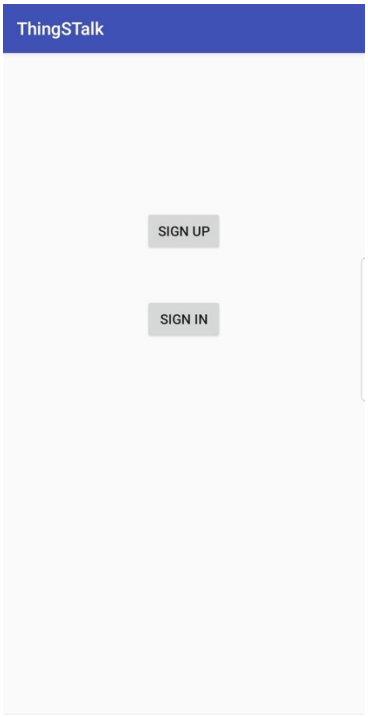

Fig. 3.41 First Page

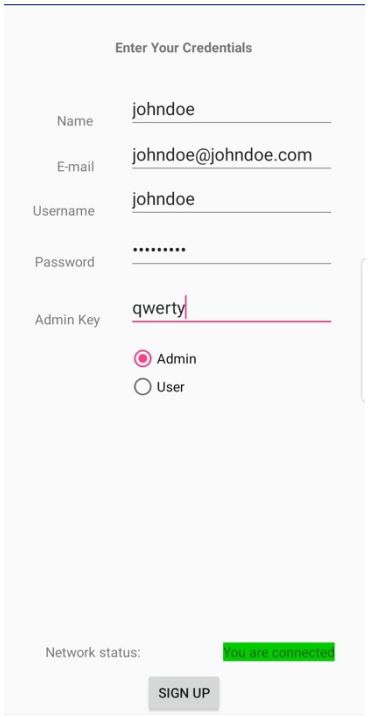

Fig. 3.42 Sign Up Page for an Admin

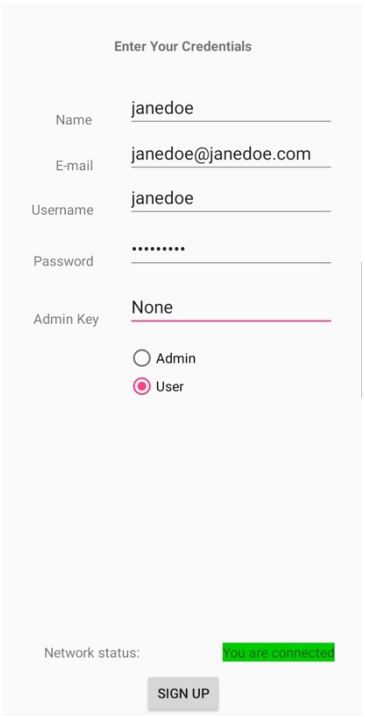

Fig. 3.43 Sign Up Page for a User

provided for the user to change his/her password and set a new one. This activity serves the purpose of both authentication as well as managing. The network status displays the status of the network manager. Since this app needs an internet connection to establish communication with the server, the network status displays whether you have connected to the internet or not. This page has been created using a relative layout.

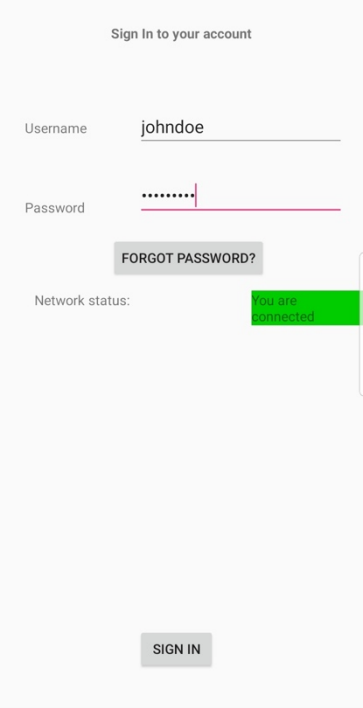

Fig. 3.44 Sign In Page

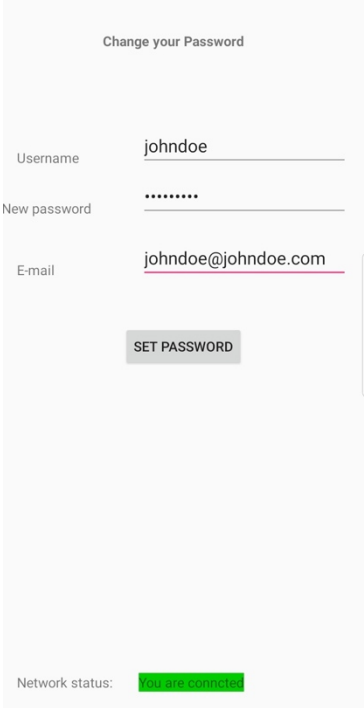

Fig. 3.45 Forgot Password Page

3.4 ACTIVITY 3:

After a successful SIGNUP or SIGN-IN session, the user is directed to a page which displays the devices pertaining to the respective user. A recycler view layout is being availed to display the devices and the corresponding router with its ID and also the status of the device (If it is ON/OFF/disconnected). This activity also enables the user to avail a drop-down menu which contains various options such as required routers he or she wants to have in their app and make the app more customized such as (Managing – add/remove users and add/remove routers) provided, the user is an ADMIN.

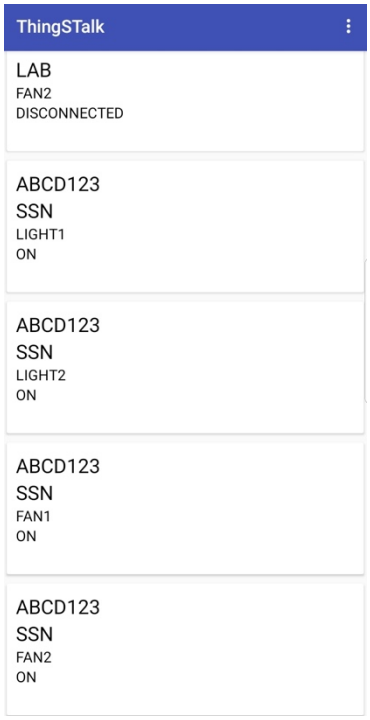

Fig. 3.46 Displaying the Various Devices with ON / OFF State

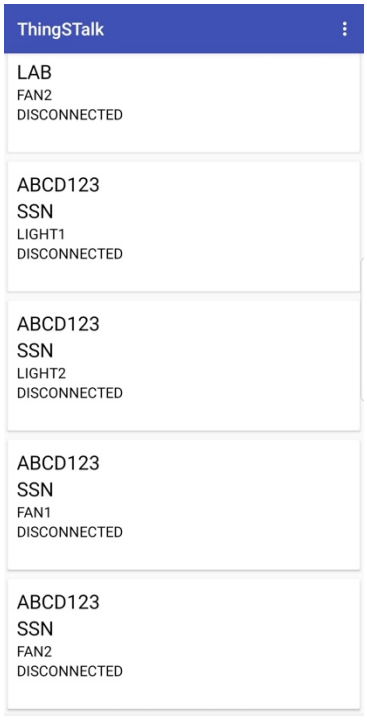

Fig. 3.47 Displaying the Various Disconnected Devices

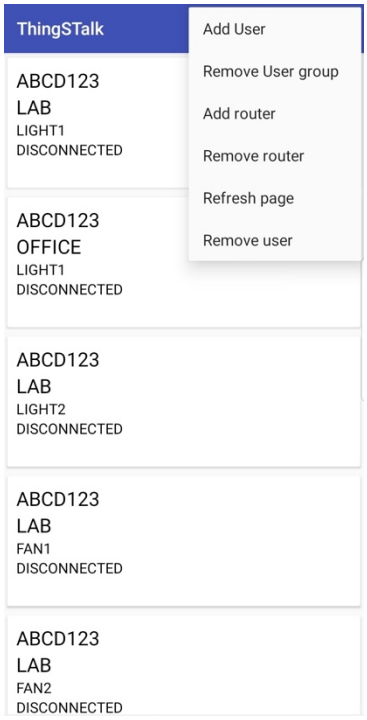

Fig. 3.48 Various Options

a) Adding and Removing Routers:

Since multiple routers can be present in multiple places of an organization, or at different rooms of a home, this functionality facilitates the user to add only the required routers he or she wants to have in their app and make the app more customized. Once, when the router is no more required, they have the facility to remove the router and also it can be replaced with another router of their choice just providing the gateway ID and its password.

b) Adding and removing users:

This functionality is brought into the app keeping in mind the need for authorizing the users. Not any person who signs up with the app can add his routers and display it. The rights for the user to display his devices and send commands is given to the user only by the admin.

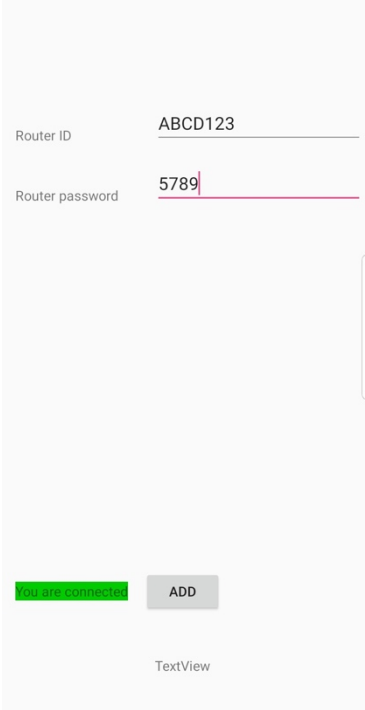

Fig. 3.49 Adding Router

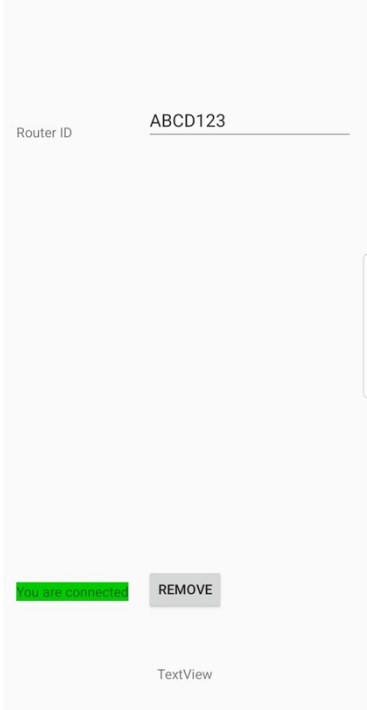

Fig. 3.50 Removing Router

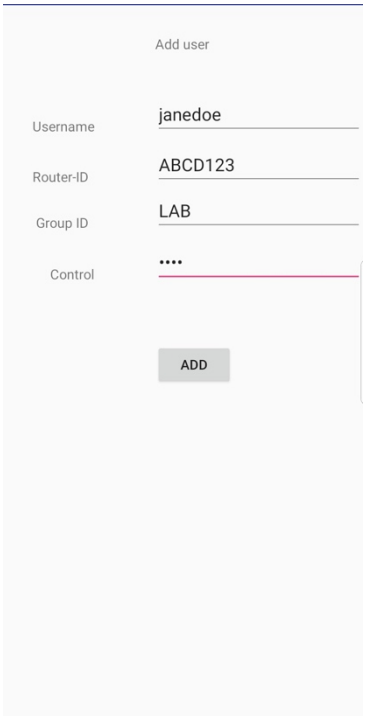

Fig. 3.51 Adding User Page

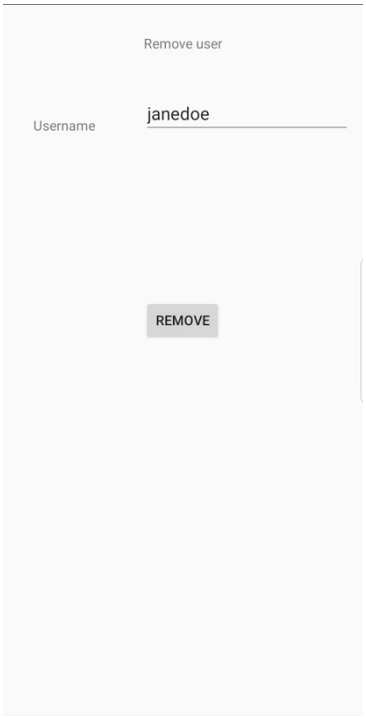

Fig. 3.52 Remove User Page

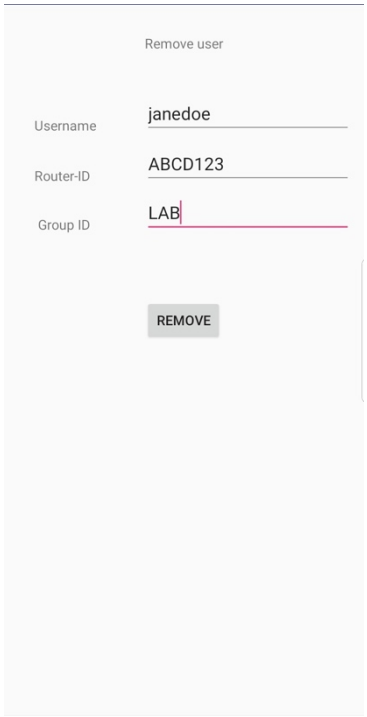

Fig. 3.53 Remove entire User Group

3.5 ACTIVITY 4:

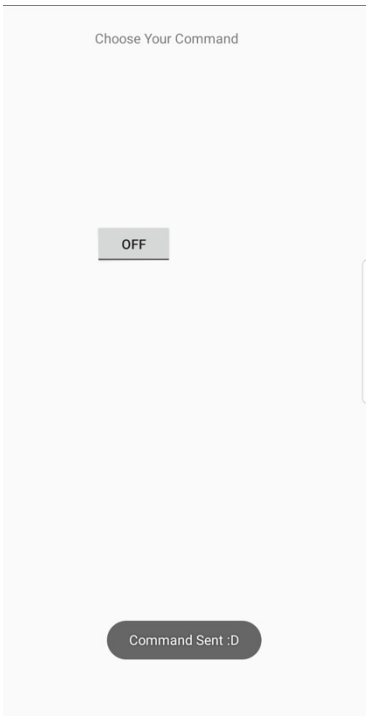

Fig. 3.54 Sending Commands

Once the particular device is clicked the control page for the device is displayed. The device can be switched ON/OFF using this page. The scope can be further extended to multiple functions such as controlling the brightness of the light or the color of light, speed of the fan etc.

3.6 CONCLUSION:

Thus, the constructed API has been integrated with the app providing the stated functionalities and hence a user-friendly interface has been developed.

CHAPTER 4
HARDWARE

4.1 INTRODUCTION:

Hardware refers to the physical components which are directed by the software to execute any command or instruction. The hardware used in this project includes a Wi-Fi-module (NodeMCU), 4 –channel Relay, a manual switch, zero watts bulb, and a 5v dc adapter. They are powered up through 5v dc adapter connected to the main supply. In our project, hardware implementation is done using the lights in our lab which are controlled remotely through the app.

4.2 COMPONENTS USED:

a) Node MCU:

Node MCU (Node Microcontroller Unit) is an open source IoT platform that is built around ESP8266 (System-on-a-chip). It is a low-cost Wi-Fi Module chip and is widely used in home automation which can be configured to connect to the Internet. NodeMCU is programmable directly through the USB port using Arduino IDE or LUA based on the eLua project and built on the ESP8266 SDK 1.4. NodeMCU is the Wi-Fi equivalent of ethernet module. It provides access to GPIO (General Purpose Input/Output) pins and the pin D0(GPIO16) can be used only for GPIO read/write. NodeMCU combines the features of a Wi-Fi access point and a microcontroller. These features make the NodeMCU extremely powerful tool for Wi-Fi networking. The input to NodeMCU is 3.3V and it is stepped up using Logic Level Converter to power up the relay board.

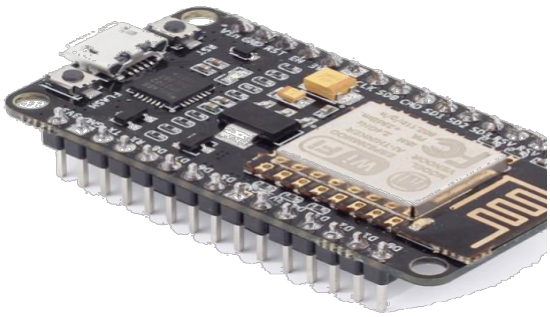

Fig. 4.1 Node MCU(ESP8266)

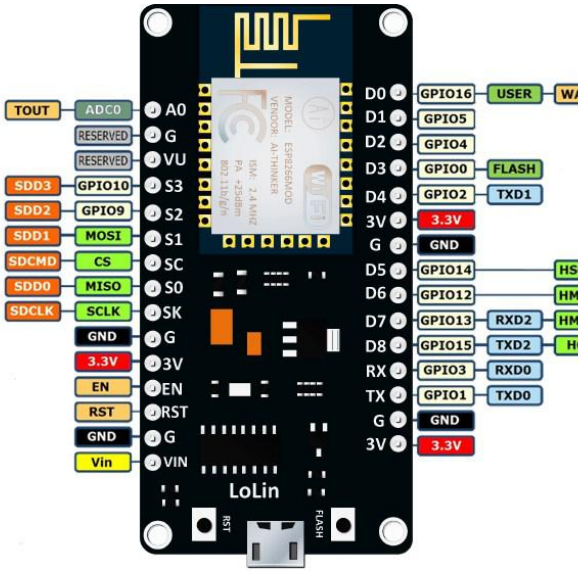

Fig. 4.2 Node MCU Pin Configuration

Courtesy: iotbytes.wordpress.com

b) 4 Channel Relays:

The 4-channel relay module makes it simple and convenient to drive loads from simple 5V digital outputs of Node MCU board or any other microcontroller. The relay is a device used for controlling a high current circuit with a low current signal. The power is supplied to the relay module via the pin labeled VCC and ground is supplied via GND pin. It is a kind of switch that can be controlled by means of a control signal. All relays have the following parts: NO (Normally open), NO (Normally closed) and COM(Common). Protection of the system is always given utmost priority in an electrical circuit. Relays are used to protect devices from various kinds of faults. They are quite common in-home appliances where there is an electrical control like turning on/off the devices. It is used for changing the state of electric devices from one to the other. There are various types of relays which are classified into various types based on the different functions they perform.

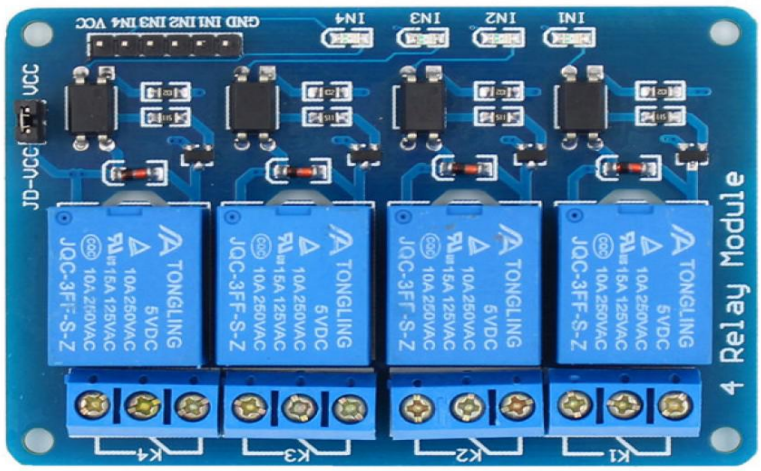

Fig. 4.3 4-channel Relay

4.3 CIRCUIT DIAGRAM:

Based on the above circuit diagram, NodeMCU and 4-channel relay are powered using a 5V DC adapter from the main supply and the lights are powered with 230V ac supply. When an on/off command comes from the app to control the lights remotely, the relay closes/opens the circuit thereby allowing the current to either switch on/off the lights in correspondence with NodeMCU input pins D1, D2, D3 and D4 accordingly.

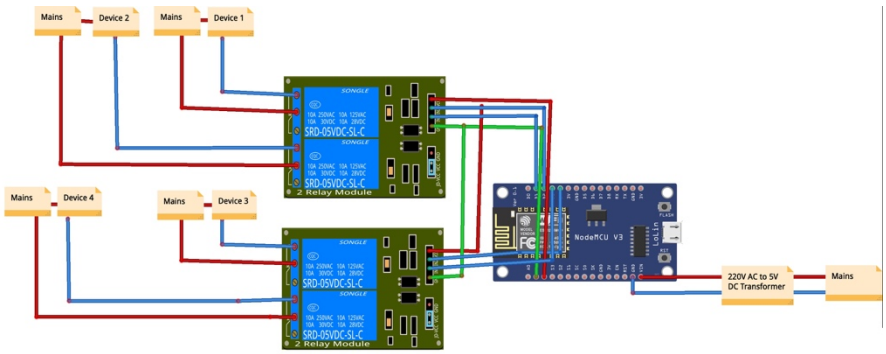

Fig. 4.4 Circuit Diagram

4.4 FLOW DIAGRAM:

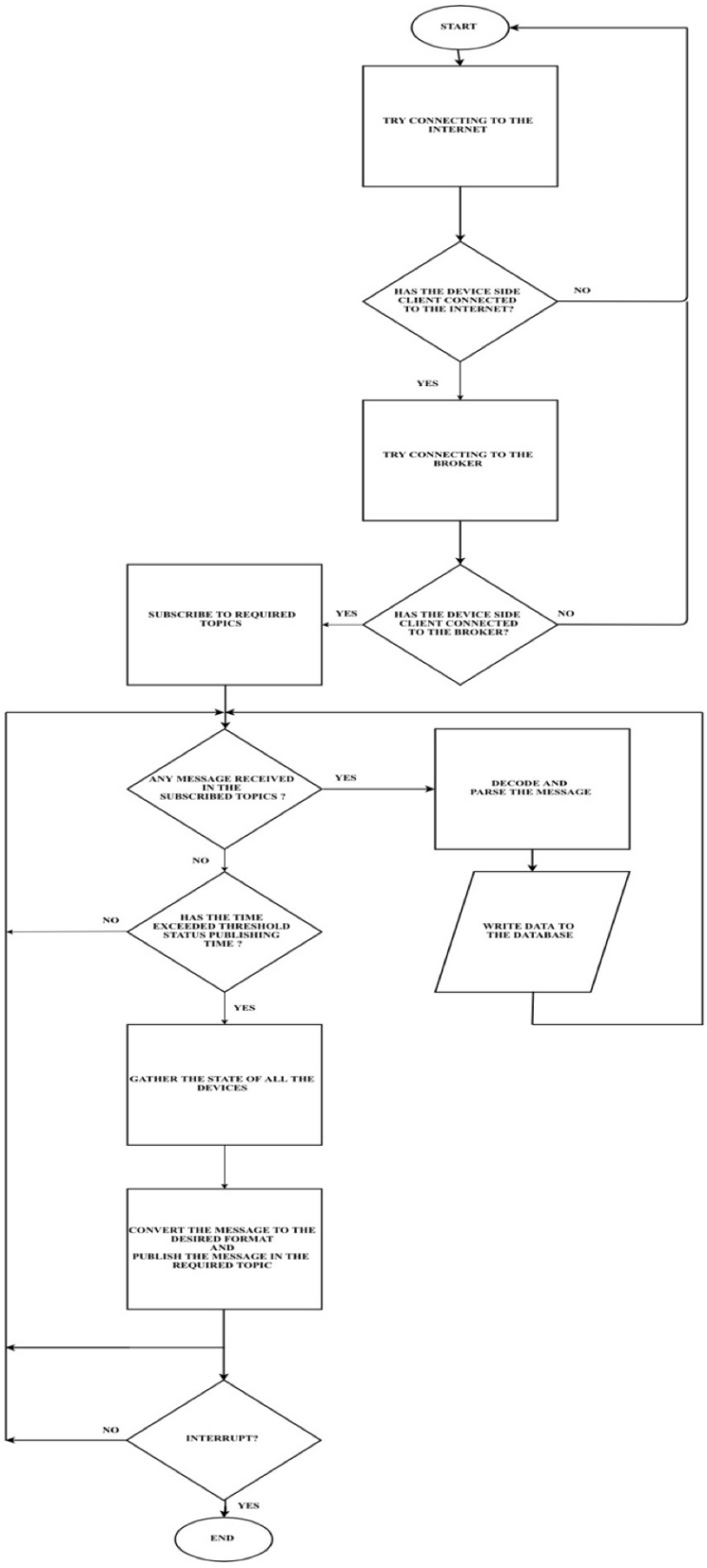

Fig. 4.5 Flow Diagram

4.5 CONCLUSION:

The detailed specifications, circuit diagram and program flow of the hardware used in the project are explained above. Thus, the implementation of hardware is achieved by remotely controlling lights using the app.

CHAPTER 5

IMPLEMENTATION AND CONCLUSION

5.1 IMPLEMENTATION AND TESTING:

A basic system using AWS at the backend and an android app to control the devices have been developed using the following components. Node MCUs for communication to and from the server, bulbs, relays and level shifters. The circuit has been developed to withstand any device with a maximum current rating of 10 Amperes. A real-time working model has been implemented availing the resource manager and a completely developed in-house model developed by Aricent Technologies (Holdings) Limited and hence the compatibility of the developed system for their devices is ensured and verified.

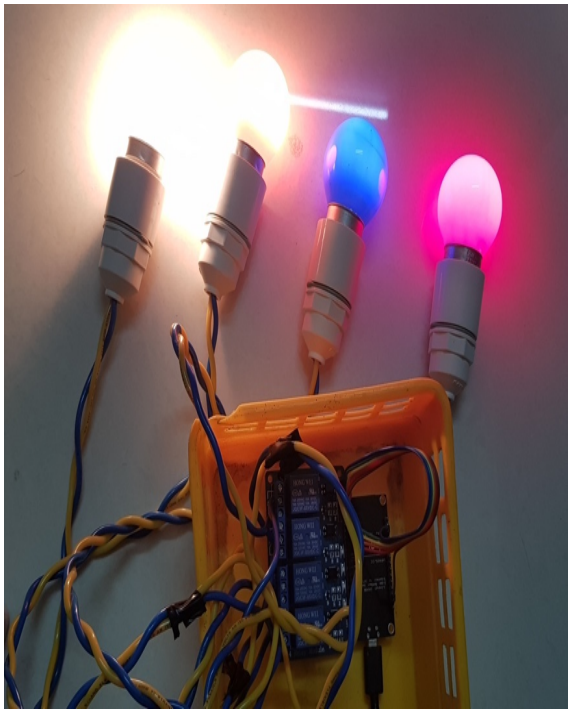

Fig. 5.1 Working Prototype

Fig. 5.2 Real Time Implementation Setup

The above setup can be used for any kind of devices with the maximum current rating of 10A.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION:

As stated in the above chapters, this project has implemented an app based Remote Home Access system that is integrated with Authentication, Authorization, Monitoring, Controlling, Managing, Logging and security as its' features. As stated in the objective, we have developed a general interface using AWS (at the backend) platform and it could further be developed and customized according to the needs of the consumer. These seven functionalities have been developed, keeping in mind the need for easy access, privacy and security of the users and overall good performance and management of the system. And hence a well-furnished system which can address all these needs has been designed, developed and implemented herewith.

6.2 FUTURE WORK:

The features can further be extended with some more add-ons such as reporting – Using push notifications to inform the user with the current state without entering the application, Fault detection-informing the user in case of any appliance mal-functioning, low voltage, short circuit, and other abnormalities. Adding more features as commands, like brightness and color control for lights, speed for fans, the temperature for Air conditioners etc. The system can be made automated by using sensors and appropriate priorities of control. It can further be integrated with machine learning by monitoring the device data and its flow continuously, to make the user experience a more customized and user-specific system.

APPENDICES

APPENDIX 1

SPECIFICATIONS OF RELAY

Trigger Voltage	5V DC
Trigger Current	70 mA
Maximum AC load current	10A @ 250V/125V
Maximum DC load current	10A @ 30V/28V (DC)
Operating Time	10 ms
Release Time	5 ms
Maximum Switching	300 operations/minute.

Table A1.1 Relay Specifications

CHAPTER 6

REFERENCES

[1] Andreas Jacobsson and Paul Davidsson (2015) 'Towards a Model of Privacy and Security for Smart Homes', IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, Italy, pp. 727 - 732

[2] Dongyu Wang, Dixon Lo, Janak Bhimani and Kazunori Sugiura (2015) 'AnyControl - IoT based Home Appliances Monitoring and Controlling', IEEE 39th Annual International Computers, Software & Applications Conference, Taichung, Taiwan, pp.487-492.

[3] Kumar Mandula, Ramu Parupalli, CH.A.S.Murty, E.Magesh, Rutul Lunagariya (2015) 'Mobile based Home Automation using Internet of Things(IoT)', International Conference on Control, Instrumentation, Communication and Computational Technologies (lCCICCT), kumaracoil, India, pp.340-343.

[4] Mahfuzur Rahman and Prabir Bhattacharya (2003) 'Remote Access And Networked Appliance Control Using Biometrics Features', IEEE Transactions on Consumer Electronics , Vol. 49, No. 2, MAY 2003

[5] Majid Al-Kuwari, Abdulrhman Ramadan, Yousef Ismael, Laith Al- Sughair, Adel Gastli (2018), 'Smart-Home Automation using IoT-based Sensing and Monitoring Platform', IEEE 12th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG), Doha, Qatar, pp.1-6.

[6] Mariana Nikolova, Frans Meijs and Peter Voorwinden (2003), 'Remote Mobile Control of Home Appliances', IEEE Transactions on Consumer Electronics, Vol. 49, No. 1, FEBRUARY 2003

[7] Pavithra.D and Ranjith Balakrishnan (2015), 'IoT based Monitoring and Control System for Home Automation', Global Conference on Communication Technologies (GCCT) , Thuckalay, India, pp.169-173.

[8] Ravi Kishore Kodali, Vishal Jain, Suvadeep Bose and Lakshmi Boppana (2016) 'IoT Based Smart Security and Home Automation System', International Conference on Computing, Communication and Automation (ICCCA), Noida, India, pp.1286-1289.

[9] Shopan Dey, Ayon Roy, Sandip Das (2016), 'Home Automation Using Internet of Thing', IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, USA, pp.1-6.