Smartly, An IoT based system

**MUHAMMAD AHSAN**

**RIZWAN AMJAD**



DEPARTMENT OF COMPUTER SCIENCES

COMSATS UNIVERSITY ISLAMABAD,

ATTOCK CAMPUS – PAKISTAN

SESSION 2018-2022

Smartly, An IoT based system

*Undertaken By:*

**MUHAMMAD AHSAN**

CIIT/FA18-BSE-028/ATK

**RIZWAN AMJAD**

CIIT/FA18-BSE-031/ATK

*Supervised By:*

**Dr. Yasir Ali Shah**

A DISSERTATION SUBMITTED AS A PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE IN

SOFTWARE ENGINEERING

DEPARTMENT OF COMPUTER SCIENCES

COMSATS UNIVERSITY ISLAMABAD,

ATTOCK CAMPUS – PAKISTAN

SESSION 2018-2022

**UNDERTAKING**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied the report entirely based on our efforts. If any part of this project is proved to be copied out from any source or found to be the reproduction of some other. We will stand by the consequences. No portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Muhammad Ahsan Rizwan Amjad

FA18-BSE-028 FA18-BSE-031

Dated: \_\_\_\_\_\_\_\_\_ Dated: \_\_\_\_\_\_\_\_

**FINAL APPROVAL**

Certified that we have read this project report submitted by Muhammad Ahsan and Rizwan Amjad and it is, in our judgment, of sufficient standard to warrant its acceptance by the Department of Computer Science, COMSATS University Islamabad, Attock Campus, for the BSSE degree.

***Committee:***

1. External Examiner \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Examiner Name

Designation

University Name

1. Supervisor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Supervisor Name

Designation

University Name

1. Head of the Department \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HoD Name

Designation

University Name

**DEDICATION**

***To our Loving Parents and Teachers***

**ACKNOWLEDGEMENT**

All praise is to Almighty Allah who bestowed upon us a minute portion of his boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Dr. Yasir Ali Shah”. Without his personal supervision, advice, and valuable guidance, the completion of this project would have been doubtful. We are deeply indebted to him for his encouragement and continual help during this work.

We are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

**PROJECT BRIEF**

PROJECT NAME Smartly, An IoT based system

ORGANIZATION NAME COMSATS

OBJECTIVE Easy to use IoT system

UNDERTAKEN BY Muhammad Ahsan

Rizwan Amjad

SUPERVISED BY Dr. Yasir Ali Shah

Lecturer

CS

COMSATS University Islamabad

STARTED ON 01/04/2021

COMPLETED ON TODO

COMPUTER USED HP Elite Book 8470p

HP Elite Book 840 G3

SOURCE LANGUAGE JavaScript,

C++

OPERATING SYSTEM Linux (Ubuntu),

Windows 10

TOOLS USED Git & GitHub,

Docker,

VSCode

PlatformIO,

Arduino IDE

**ABSTRACT**

The problem with the currently available IoT (Internet of Things) systems is that they require a lot of professional skills to build and deploy. It is very difficult for laymen to set up their IoT network. There are many IoT-related solutions available in the market but they are mainly focused on cloud-based services for IoT. There are very limited solutions available for IoT hardware and most of them require detailed technical knowledge of programming and electronics. To solve most of the above-mentioned problems we are going to design and develop a complete standalone IoT system. This system will include wirelessly connected plug and play IoT hardware devices, cloud-based services for managing and storing data from IoT devices, and a cross-platform Mobile App for controlling and monitoring IoT devices.

**TABLE OF CONTENTS**

1. [INTRODUCTION 2](#_Toc102815605)

[1.1 Introduction 2](#_Toc102815606)

[1.2 Problem Statement 2](#_Toc102815607)

[1.3 Proposed Solution 2](#_Toc102815608)

[1.4 Motivation 2](#_Toc102815609)

[1.5 Tools 2](#_Toc102815610)

[1.6 Frameworks 2](#_Toc102815611)

[1.7 IoT Boards 2](#_Toc102815612)

[1.8 Goals 2](#_Toc102815613)

[1.9 Objectives 2](#_Toc102815614)

1. [LITERATURE REVIEW 2](#_Toc102815615)

[2.1 Introduction 2](#_Toc102815616)

[2.2 Existing Systems 2](#_Toc102815617)

[2.2.1 Blynk 2](#_Toc102815618)

[2.2.2 Google Cloud IoT Core 2](#_Toc102815619)

[2.2.3 ThingSpeak 2](#_Toc102815620)

[2.3 Proposed System 2](#_Toc102815621)

[2.3.1 IoT Hardware 2](#_Toc102815622)

[2.3.2 Cloud-Based Service 2](#_Toc102815623)

[2.3.3 Cross-platform Mobile App 2](#_Toc102815624)

[2.4 Comparison 2](#_Toc102815625)

1. [REQUIREMENT ANALYSIS 2](#_Toc102815626)

[3.1 Introduction 2](#_Toc102815627)

[3.2 Functional Requirements (Specifications) 2](#_Toc102815628)

[3.1.1 Mobile App 2](#_Toc102815629)

[3.1.2 IoT Hardware 2](#_Toc102815630)

[3.3 Functional Requirements (Analysis) 2](#_Toc102815631)

[3.4 Non Functional Requirements 2](#_Toc102815632)

[3.4.1 Availability 2](#_Toc102815633)

[3.4.2 Interoperability 2](#_Toc102815634)

[3.4.3 Performance 2](#_Toc102815635)

[3.4.4 Privacy 2](#_Toc102815636)

[3.4.5 Recoverability 2](#_Toc102815637)

[3.4.6 Reliability 2](#_Toc102815638)

[3.4.7 Scalability 2](#_Toc102815639)

[3.4.8 Supportability 2](#_Toc102815640)

[3.4.9 Usability 2](#_Toc102815641)

1. [SYSTEM DESIGN 2](#_Toc102815642)

[4.1 Introduction 2](#_Toc102815643)

[4.2 Use Case Diagram 2](#_Toc102815644)

[4.2.1 Mobile App Use Case Diagram 2](#_Toc102815645)

[4.2.2 Gateway Use Case Diagram 2](#_Toc102815646)

[4.3 Activity Diagram 2](#_Toc102815647)

[4.3.1 Mobile App Activity Diagram 2](#_Toc102815648)

[4.3.2 Gateway Activity Diagram 2](#_Toc102815649)

[4.3.3 Sensor Node Activity Diagram 2](#_Toc102815650)

[4.3.4 Actuator Node Activity Diagram 2](#_Toc102815651)

[4.4 Sequence Diagram 2](#_Toc102815652)

[4.5 Component Diagram 2](#_Toc102815653)

[4.5.1 Cloud-Based Service Component Diagram 2](#_Toc102815654)

[4.5.2 Gateway Component Diagram 2](#_Toc102815655)

[4.5.3 Sensor Node Component Diagram 2](#_Toc102815656)

[4.5.4 Actuator Node Component Diagram 2](#_Toc102815657)

[4.6 Deployment Diagram 2](#_Toc102815658)

[4.7 Entity Relationship Diagram 2](#_Toc102815659)

1. [IMPLEMENTATION 2](#_Toc102815660)

[5.1 Introduction 2](#_Toc102815661)

[5.2 Mobile App UI (Android) 2](#_Toc102815662)

[5.2.1 Splash Screen 2](#_Toc102815663)

[5.2.2 Login Screen 2](#_Toc102815664)

[5.2.3 Register Screen 2](#_Toc102815665)

[5.2.4 Home Screen 2](#_Toc102815666)

[5.2.5 Account Screen 2](#_Toc102815667)

[5.2.6 Sensors Screen 2](#_Toc102815668)

[5.2.7 Actuator Screen 2](#_Toc102815669)

[5.2.8 QR-Code Scan Screen 2](#_Toc102815670)

[5.2.9 Analytics Screen 2](#_Toc102815671)

[5.3 Mobile App UI (iOS) 2](#_Toc102815672)

[5.3.1 Splash Screen 2](#_Toc102815673)

[5.3.2 Login Screen 2](#_Toc102815674)

[5.3.3 Register Screen 2](#_Toc102815675)

[5.3.4 Home Screen 2](#_Toc102815676)

[5.3.5 Account Screen 2](#_Toc102815677)

[5.3.6 Sensors Screen 2](#_Toc102815678)

[5.3.7 Actuator Screen 2](#_Toc102815679)

[5.3.8 Qr-Code Scan Screen 2](#_Toc102815680)

[5.3.9 Analytics Screen 2](#_Toc102815681)

[5.4 Gateway Config Portal UI 2](#_Toc102815682)

[5.4.1 Login Screen 2](#_Toc102815683)

[5.4.2 Wi-Fi Screen 2](#_Toc102815684)

[5.4.3 Account Screen 2](#_Toc102815685)

[5.4.4 Security Screen 2](#_Toc102815686)

[5.5 Gateway 2](#_Toc102815687)

[5.6 Sensor Nodes 2](#_Toc102815688)

[5.6.1 Temperature Sensor 2](#_Toc102815689)

[5.6.2 Barometric Pressure Sensor 2](#_Toc102815690)

[5.7 Actuator Node 2](#_Toc102815691)

1. [TESTING 2](#_Toc102815692)

[6.1 Introduction 2](#_Toc102815693)

[6.2 App Testing 2](#_Toc102815694)

[6.3 Cloud-Based Service Testing 2](#_Toc102815695)

[6.4 Hardware Testing 2](#_Toc102815696)

[6.5 Integration Testing 2](#_Toc102815697)

[6.6 System Testing 2](#_Toc102815698)

[6.7 Test Cases 2](#_Toc102815699)

[6.7.1 Mobile App: Register 2](#_Toc102815700)

[6.7.2 Mobile App: Login 2](#_Toc102815701)

[6.7.3 Mobile App: Gateways 2](#_Toc102815702)

[6.7.4 Mobile App: Sensors 2](#_Toc102815703)

[6.7.5 Mobile App: Actuators 2](#_Toc102815704)

[6.7.6 Mobile App: QR Code Scan 2](#_Toc102815705)

[6.7.7 Mobile App: Account screen 2](#_Toc102815706)

1. [CONCLUSION AND FUTURE WORK 2](#_Toc102815707)

[7.1 Conclusion 2](#_Toc102815708)

[7.2 Future Work 2](#_Toc102815709)

1. [REFERENCES 2](#_Toc102815710)

**List of Figures**

[Figure 1.1 Illustration of Smartly system 2](#_Toc102815757)

[Figure 4.1 System Use Case diagram 2](#_Toc102815758)

[Figure 4.2 Mobile App Use Case diagram 2](#_Toc102815759)

[Figure 4.3 Gateway Use Case diagram 2](#_Toc102815760)

[Figure 4.4 Mobile App Activity diagram 2](#_Toc102815761)

[Figure 4.5 Gateway Activity diagram 2](#_Toc102815762)

[Figure 4.6 Sensor Node Activity diagram 2](#_Toc102815763)

[Figure 4.7 Actuator Node Activity diagram 2](#_Toc102815764)

[Figure 4.8 System Sequence diagram 2](#_Toc102815765)

[Figure 4.9 System Component diagram 2](#_Toc102815766)

[Figure 4.10 Cloud-based service Component diagram 2](#_Toc102815767)

[Figure 4.11 Gateway Component diagram 2](#_Toc102815768)

[Figure 4.12 Sensor Node Component diagram 2](#_Toc102815769)

[Figure 4.13 Actuator Node Component diagram 2](#_Toc102815770)

[Figure 4.14 System Deployment diagram 2](#_Toc102815771)

[Figure 4.15 Entity Relationship Diagram 2](#_Toc102815772)

[Figure 5.1 Splash Screen on Android 2](#_Toc102815773)

[Figure 5.2 Login Screen on Android 2](#_Toc102815774)

[Figure 5.3 Register Screen on Android 2](#_Toc102815775)

[Figure 5.4 Home Screen on Android 2](#_Toc102815776)

[Figure 5.5 Account Screen on Android 2](#_Toc102815777)

[Figure 5.6 Sensor Screen on Android 2](#_Toc102815778)

[Figure 5.7 Actuator Screen on Android 2](#_Toc102815779)

[Figure 5.8 QR-Code Scan Screen on Android 2](#_Toc102815780)

[Figure 5.9 Analytic Screen on Android 2](#_Toc102815781)

[Figure 5.10 Splash Screen on iOS 2](#_Toc102815782)

[Figure 5.11 Login Screen on iOS 2](#_Toc102815783)

[Figure 5.12 Register Screen on iOS 2](#_Toc102815784)

[Figure 5.13 Home Screen on iOS 2](#_Toc102815785)

[Figure 5.14 Account Screen on iOS 2](#_Toc102815786)

[Figure 5.15 Sensor Screen on iOS 2](#_Toc102815787)

[Figure 5.16 Actuator Screen on iOS 2](#_Toc102815788)

[Figure 5.17 QR-Code Scan Screen on iOS 2](#_Toc102815789)

[Figure 5.18 Analytic Screen on iOS 2](#_Toc102815790)

[Figure 5.19 Gateway Config App Login Screen 2](#_Toc102815791)

[Figure 5.20 Gateway Config App Wi-Fi Screen 2](#_Toc102815792)

[Figure 5.21 Gateway Config App Account Screen 2](#_Toc102815793)

[Figure 5.22 Gateway Config App Security Screen 2](#_Toc102815794)

**List of Tables**

[Table 2.1 Comparison of systems 2](#_Toc102815795)

[Table 3.1 Cross-platform support 2](#_Toc102815796)

[Table 3.2 Create account 2](#_Toc102815797)

[Table 3.3 Login 2](#_Toc102815798)

[Table 3.4 Unique key 2](#_Toc102815799)

[Table 3.5 Listing gateways 2](#_Toc102815800)

[Table 3.6 View sensor nodes 2](#_Toc102815801)

[Table 3.7 View actuator nodes 2](#_Toc102815802)

[Table 3.8 View the current state of the actuator 2](#_Toc102815803)

[Table 3.9 Latest reading of sensor node 2](#_Toc102815804)

[Table 3.10 Sensor analytics 2](#_Toc102815805)

[Table 3.11 Ready to use IoT hardware 2](#_Toc102815806)

[Table 3.12 Plug and play hardware 2](#_Toc102815807)

[Table 3.13 IoT gateway 2](#_Toc102815808)

[Table 3.14 Sensor nodes 2](#_Toc102815809)

[Table 3.15 Actuator nodes 2](#_Toc102815810)

[Table 3.16 Configurable gateway 2](#_Toc102815811)

[Table 3.17 Adding sensor nodes to the gateway 2](#_Toc102815812)

[Table 3.18 Adding actuator nodes to the gateway 2](#_Toc102815813)

[Table 3.19 Connecting gateway to account 2](#_Toc102815814)

[Table 3.20 Connecting gateway to the internet 2](#_Toc102815815)

[Table 4.1 System Use Case description 2](#_Toc102815816)

[Table 4.2 Mobile App Use Case description 2](#_Toc102815817)

[Table 4.3 Mobile App Use Case description 2](#_Toc102815818)

[Table 6.1 Test cases for Register Screen 2](#_Toc102815819)

[Table 6.2 Test cases for Login Screen 2](#_Toc102815820)

[Table 6.3 Test cases for Gateway Screen 2](#_Toc102815821)

[Table 6.4 Test cases for Sensor Screen 2](#_Toc102815822)

[Table 6.5 Test cases for Actuator Screen 2](#_Toc102815823)

[Table 6.6 Test cases for QR Code Scanner Screen 2](#_Toc102815824)

[Table 6.7 Test cases for Account Screen 2](#_Toc102815825)

**LIST OF ABBREVIATIONS**

API Application Programming Interface

AP Access Point

HTTP Hypertext Transfer Protocol

IoT Internet of Things

IP Internet Protocol

MQTT Message Queuing Telemetry Transport

MAC Medium Access Control

REST Representational State Transfer

Wi-Fi Wireless Fidelity

Chapter 1

# Introduction

* 1. Introduction

The Internet of Things (IoT) is going to revolutionize our future. According to some estimates, there will be 80 billion devices connected to the cloud by 2025 [1]. This huge amount of connected devices is going to provide us the opportunity to grow with this market. Different organizations out there are inclining towards automation and IoT is playing an important role.

Many companies are providing their solution for IoT, Some are providing cloud-based services and some are providing hardware solutions. But to use those solutions user needs to have good technical knowledge. It is very difficult for any layman to automate his devices using available technologies.

We are going to design and develop a complete standalone IoT system. This system will include wirelessly connected plug-and-play IoT hardware devices, a cloud-based service for managing and storing data from IoT devices, and a cross-platform Mobile App for controlling and monitoring IoT devices.

We are going to name our system ***“Smartly”*** with the motto of ***“IoT for Everyone”***.

* 1. Problem Statement

The main problem with the currently available IoT solutions is that they are mainly focusing on providing cloud-based services and they are not focusing on developing IoT hardware devices which is the main limitation of all available solutions. Most of the available IoT Hardware (Microcontrollers, Sensors, Actuators, and IoT boards) is not available in a directly useable form. The user needs to have a good knowledge of Bare Metal Programming and Electronics to set up his own IoT network. Currently, in most of the available IoT devices, the sensors and actuators are directly connected with the microcontroller through a wired medium and the same microcontroller is also responsible for sending data to the server. This will bound users to a fixed system that cannot be changed easily.

* 1. Proposed Solution

To tackle all of the above-mentioned problems, we are going to develop a new standalone IoT system that will include wirelessly connected plug-and-play IoT hardware devices, a cloud-based service for managing and storing data from IoT devices, and a cross-platform Mobile App for controlling and monitoring IoT devices. The main concern of our project is to build IoT hardware that can easily be configured by a layman through the built-in web interface of the device. The user just needs to connect his Smart Phone or Laptop to the IoT device through Wi-Fi, then by using the browser he will open the given IP address of the device and it will open a web interface through which the user can configure the system. In this project, multiple IoT devices will be built which include an IoT gateway, IoT sensors, and IoT actuators. All of these sensors and actuators will communicate with the IoT gateway wirelessly and the IoT gateway will transfer the data to the cloud-based service. Figure 1.1 illustrates the working of the proposed system.

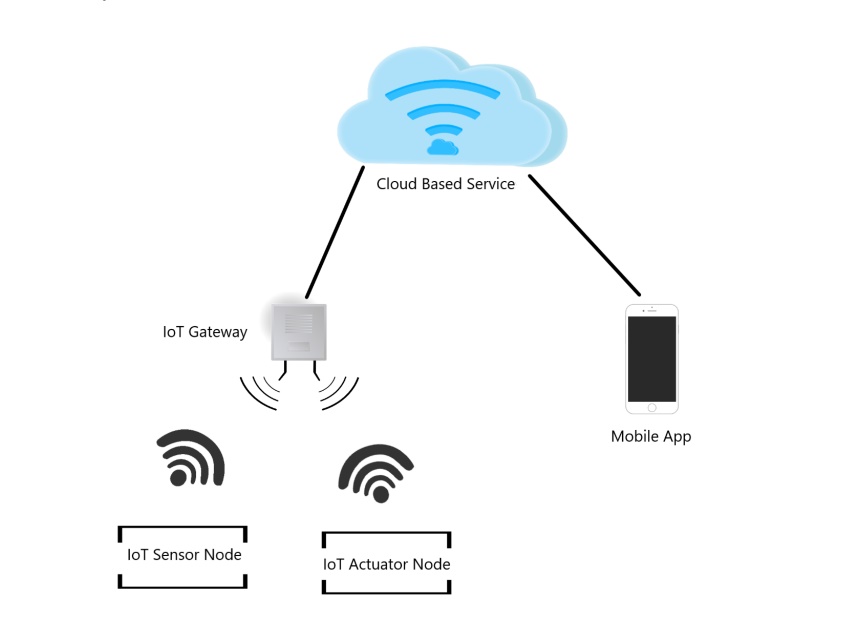


Figure 1.1 Illustration of Smartly system

* 1. Motivation

The main motivation behind this project is the 4th Industrial Revolution (4IR) [2]. IoT is a part of the 4IR and makes people's life easier. Controlling multiple devices from one platform easily, effectively, efficiently, and quickly, motivates us to build a complete and standalone system for the Internet of Things.

* 1. Tools
* VS Code
* PlatformIO
* Arduino IDE
* Postman
* Docker
* Git and GitHub
* MQTTLens
* MongoDB Compass
* Node JS
* Star UML
  1. Frameworks
* React Native
* Arduino
* Express JS
  1. IoT Boards
* ESP 01S (ESP8266)
* ESP 12E/F (ESP8266)
* ESP 32
  1. Goals
* Our long-term goal is to create a new IoT system that will ease the use and deployment of IoT systems.
* To contribute and cover the emerging market of IoT.
* To start a company based on this project which will provide automation services to people.
* To make “Smartly” a globally recognized system.
  1. Objectives
* To build an IoT gateway that can easily be configured by a layman through the built-in web interface.
* To build an IoT gateway that will manage and connect all the sensor and actuator nodes to the cloud-based service.
* To build multiple sensor nodes for monitoring the environment (like Temperature, Humidity, Pressure, Air-Quality, etc.).
* To build an actuator node for controlling appliances (like Lights, Fan, etc.).
* To build a complete and new cloud-based service for IoT devices through which users can communicate with their IoT system.
* To provide the user with an interface for controlling and monitoring IoT devices

Chapter 2

# Literature Review

* 1. Introduction

In the literature review first of all we will discuss the available solutions for IoT in the market. Also, we’ll discuss their functionalities and limitations. In the next section, we will discuss the proposed system. We also discuss the main parts of the proposed system and its functionalities. In the last section, we will compare the existing systems with the proposed system.

* 1. Existing Systems

There are multiple IoT systems and services available in the market. Here we discuss some of the most used IoT systems and also their limitations.

* + 1. Blynk

Blynk is an IoT platform, it provides services to developers and companies for managing and controlling their IoT systems [3]. Blynk consists of a cloud-based service and mobile app.

The user who wants to automate his devices using Blynk requires to install the app and create an account. They provide an API token through which users can connect their hardware devices. These are the limitations of the Blynk system

* The user needs to build/program his hardware.
* The user needs to configure the Blynk app according to hardware.
* Minor difference in the configuration of the app and hardware creates major problems.
* It requires professional developers to automate devices.
  + 1. Google Cloud IoT Core

Cloud IoT Core is the fully managed IoT service provided by Google. It consists of cloud-based services and a web dashboard for monitoring and controlling IoT devices [4]. These are some limitations of the Cloud IoT core system.

* No dedicated mobile app for controlling and monitoring IoT hardware.
* No ready-to-use hardware.
* Users need to have good knowledge of API keys, MQTT, Web Sockets, and HTTP to use Cloud IoT Core services.
* Need to use third party IoT hardware or custom-built IoT hardware
  + 1. ThingSpeak

ThingSpeak is a cloud-based server for IoT data collection. It provides user services to connect their IoT hardware with them by using an API key. ThingSpeak provides different visualization methods for sensor data [5]. There are different types of charts and graphs available for the visualization of the data. These are some important limitations ThingSpeak.

* Only web-based dashboard and no official mobile app for the user.
* Users need to have good knowledge of IoT systems to use the ThingSpeak service.
* User needs to use third-party IoT hardware or custom-built IoT hardware.
  1. Proposed System

The proposed system will be a complete standalone IoT system. This system will include wirelessly connected plug-and-play IoT hardware devices, cloud-based services for managing and storing data from IoT devices, and a cross-platform Mobile App for controlling and monitoring these IoT devices. The system will be divided mainly into three parts. The details are given below:

* + 1. IoT Hardware

The hardware part of the “Smartly” is mainly divided into three parts. That is an IoT gateway, sensor node, and actuator node. Details of each part are given below.

* **IoT Gateway**

The work of the IoT gateway in the system is to manage all the available wireless sensors and actuators. This gateway is also connected to a cloud-based server. This gateway collect data from all available sensors and send it to the server. Other than that, it also maps incoming actuator's controls request to respective IoT actuators node connected with it. Our IoT gateway is based on ESP32 and ESP12 IoT boards.

* **Sensor Node**

Sensor nodes are used to collect data from the environment (like Temperature, Humidity, Pressure, Air-Quality, etc.) and send it to the IoT gateway. Sensor nodes are based on ESP12 E/F or ESP01S boards depending on the sensor type.

* **Actuator Node**

Actuator nodes are responsible for controlling the different appliances. They are just like a button controlled using IoT. Actuator nodes are connected to the IoT gateway. Actuator nodes are based on ESP01 IoT boards.

* + 1. Cloud-Based Service

Cloud-based service in our IoT system will manage all the IoT gateways which are connected with it. It also stores data coming from different sensors. This cloud-based service also provides the interface to a mobile app for controlling and monitoring purposes. This cloud-based service also maps the incoming actuator's control request to desired IoT gateway.

* + 1. Cross-platform Mobile App

The mobile app is used to monitor and control IoT devices. Users can view their sensors data in the app and they can also view the analytics based on the previous data of sensors stored on the cloud-based service. Users can also control their appliances with this app in real-time. We will use React Native to build this cross-platform app.

* 1. Comparison

Table 2.1 Comparison of systems

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Services** | **Smartly** | **Blynk** | **Cloud IoT Core** | **ThingSpeak** |
| Mobile App | ✓ | ✓ | **🗙** | **🗙** |
| Cloud-based Service | ✓ | ✓ | ✓ | ✓ |
| IoT gateway | ✓ | **🗙** | **🗙** | **🗙** |
| IoT sensor nodes | ✓ | **🗙** | **🗙** | **🗙** |
| IoT actuator nodes | ✓ | **🗙** | **🗙** | **🗙** |

Chapter 3

# Requirement Analysis

* 1. Introduction

In this chapter, we will discuss the user and system requirements of the “Smartly” system. The first section of this chapter will discuss the functional requirements of the system. The second section explains in detail the non-functional requirements of the system.

* 1. Functional Requirements (Specifications)
     1. Mobile App
* The Mobile App should run on both Android and iOS.
* The user must be able to create an account.
* The user must be able to login into the system.
* The user must be able to get a unique key for connecting with the IoT hardware.
* The user should be able to view all the gateways connected with their account.
* The user should be able to view all the sensor nodes of the selected gateway.
* The user should be able to view all the actuator nodes of the selected gateway.
* The user should be able to view the current state of an actuator node.
* The user should be able to view the latest reading received from a sensor node.
* The user could be able to view sensor analytics.
  + 1. IoT Hardware
* There must be ready-to-use IoT hardware.
* IoT hardware must be plug-and-play.
* The system must have an IoT gateway.
* The system must have sensor nodes.
* The system must have actuator nodes.
* The IoT gateway must be configurable using the web interface.
* Users must be able to add sensor nodes to the gateway using the web interface.
* Users must be able to add actuator nodes to the gateway using the web interface.
* The user must be able to connect the gateway to the account using the unique
* The gateway must be able to connect to the network using Wi-Fi.
  1. Functional Requirements (Analysis)

Table 3.1 Cross-platform support

|  |  |
| --- | --- |
| **Identifier** | **FR-1** |
| **Title** | Cross platform support |
| **Requirement** | The Mobile App should run on both Android and iOS. |
| **Source** | Users |
| **Rationale** | -- |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | Medium |

Table 3.2 Create account

|  |  |
| --- | --- |
| **Identifier** | **FR-2** |
| **Title** | Create account |
| **Requirement** | The user must be able to create an account. |
| **Source** | Users |
| **Rationale** | This app requires authorization of the user to control their hardware |
| **Restrictions and Risk** | The user must have a valid email account |
| **Dependencies** | -- |
| **Priorities** | High |

Table 3.3 Login

|  |  |
| --- | --- |
| **Identifier** | **FR-3** |
| **Title** | Login |
| **Requirement** | The user must be able to login into the system. |
| **Source** | Users |
| **Rationale** | Same as FR-2 |
| **Restrictions and Risk** | The user must have an account |
| **Dependencies** | FR-2 |
| **Priorities** | High |

Table 3.4 Unique key

|  |  |
| --- | --- |
| **Identifier** | **FR-4** |
| **Title** | Unique key |
| **Requirement** | The user must be able to get a unique key for connecting with the IoT hardware. |
| **Source** | Users |
| **Rationale** | This key will serve the purpose of authentication with the account in the IoT gateways. |
| **Restrictions and Risk** | -- |
| **Dependencies** | FR-3 |
| **Priorities** | High |

Table 3.5 Listing gateways

|  |  |
| --- | --- |
| **Identifier** | **FR-5** |
| **Title** | Listing gateways |
| **Requirement** | The user should be able to view all the sensor nodes of the selected gateway. |
| **Source** | Users |
| **Rationale** | The user will be able to navigate to the Sensors and Actuators of that Particular Gateway. |
| **Restrictions and Risk** | -- |
| **Dependencies** | FR-3 |
| **Priorities** | Medium |

Table 3.6 View sensor nodes

|  |  |
| --- | --- |
| **Identifier** | **FR-6** |
| **Title** | View sensor nodes |
| **Requirement** | The user should be able to view all the sensor nodes of the selected gateway. |
| **Source** | Users |
| **Rationale** | The user can check the latest reading. And navigate to analytics for a particular sensor |
| **Restrictions and Risk** | -- |
| **Dependencies** | FR-3 |
| **Priorities** | Medium |

Table 3.7 View actuator nodes

|  |  |
| --- | --- |
| **Identifier** | **FR-7** |
| **Title** | View actuator nodes |
| **Requirement** | The user should be able to view all the actuator nodes of the selected gateway. |
| **Source** | Users |
| **Rationale** | The user can check the actuator status and navigate to a particular actuator to control it. |
| **Restrictions and Risk** | -- |
| **Dependencies** | FR-3 |
| **Priorities** | High |

Table 3.8 View the current state of the actuator

|  |  |
| --- | --- |
| **Identifier** | **FR-8** |
| **Title** | View the current state of the actuator |
| **Requirement** | The user should be able to view the current state of the actuator node. |
| **Source** | Users |
| **Rationale** | So that users can check and monitor without opening it. |
| **Restrictions and Risk** | -- |
| **Dependencies** | FR-3 |
| **Priorities** | Medium |

Table 3.9 Latest reading of sensor node

|  |  |
| --- | --- |
| **Identifier** | **FR-9** |
| **Title** | The latest reading of sensor node |
| **Requirement** | The user should be able to view the latest reading received from a sensor node. |
| **Source** | Users |
| **Rationale** | Same as FR-8 |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | Medium |

Table 3.10 Sensor analytics

|  |  |
| --- | --- |
| **Identifier** | **FR-10** |
| **Title** | Sensor analytics |
| **Requirement** | The user could be able to view sensor analytics. |
| **Source** | Users |
| **Rationale** | They will be able to check for the previous readings. |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | Medium |

Table 3.11 Ready to use IoT hardware

|  |  |
| --- | --- |
| **Identifier** | **FR-11** |
| **Title** | Ready to use IoT hardware. |
| **Requirement** | There must be ready-to-use IoT hardware. |
| **Source** | Users |
| **Rationale** | The user doesn’t have to worry about programming the hardware. |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | High |

Table 3.12 Plug and play hardware

|  |  |
| --- | --- |
| **Identifier** | **FR-12** |
| **Title** | Plug and play hardware |
| **Requirement** | IoT hardware must be plug-and-play. |
| **Source** | Users |
| **Rationale** | Same as FR-11,  The user doesn’t need to worry about making connections of hardware once the connection is lost. |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | High |

Table 3.13 IoT gateway

|  |  |
| --- | --- |
| **Identifier** | **FR-13** |
| **Title** | IoT gateway |
| **Requirement** | The system must have an IoT gateway. |
| **Source** | Users |
| **Rationale** | -- |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | High |

Table 3.14 Sensor nodes

|  |  |
| --- | --- |
| **Identifier** | **FR-14** |
| **Title** | Sensor Nodes |
| **Requirement** | The system must have sensor nodes. |
| **Source** | Users |
| **Rationale** | -- |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | High |

Table 3.15 Actuator nodes

|  |  |
| --- | --- |
| **Identifier** | **FR-15** |
| **Title** | Actuator nodes |
| **Requirement** | The system must have actuator nodes. |
| **Source** | Users |
| **Rationale** | -- |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | Medium |

Table 3.16 Configurable gateway

|  |  |
| --- | --- |
| **Identifier** | **FR-16** |
| **Title** | Configurable gateway |
| **Requirement** | The IoT gateway must be configurable using the web interface. |
| **Source** | Users |
| **Rationale** | So the user doesn’t have to explicitly program the hardware. |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | High |

Table 3.17 Adding sensor nodes to the gateway

|  |  |
| --- | --- |
| **Identifier** | **FR-17** |
| **Title** | Adding sensor nodes to the gateway |
| **Requirement** | Users must be able to add sensor nodes to the gateway using the web interface. |
| **Source** | Users |
| **Rationale** | Same as FR-16 |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | Medium |

Table 3.18 Adding actuator nodes to the gateway

|  |  |
| --- | --- |
| **Identifier** | **FR-18** |
| **Title** | Adding actuator nodes to the gateway |
| **Requirement** | Users must be able to add actuator nodes to the gateway using the web interface. |
| **Source** | Users |
| **Rationale** | Same as FR-16 |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | High |

Table 3.19 Connecting gateway to account

|  |  |
| --- | --- |
| **Identifier** | **FR-19** |
| **Title** | Connecting gateway to account |
| **Requirement** | The user must be able to connect the gateway to the account using the unique key. |
| **Source** | Users |
| **Rationale** | Same as FR-16 |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | High |

Table 3.20 Connecting gateway to the internet

|  |  |
| --- | --- |
| **Identifier** | **FR-20** |
| **Title** | Connecting gateway to the internet |
| **Requirement** | The gateway must be able to connect to the network using Wi-Fi. |
| **Source** | Users |
| **Rationale** | -- |
| **Restrictions and Risk** | -- |
| **Dependencies** | -- |
| **Priorities** | High |

* 1. Non Functional Requirements
     1. Availability
* The system must be available 24/7. Since this is an IoT-based system so if the cloud-based service goes down, the whole system will be down.
  + 1. Interoperability
* The system is divided into many small components, so each component should be able to communicate with the other without any issue.
  + 1. Performance
* The system must be able to handle 10 thousand concurrent requests at a single time.
* The system must be able to change the actuator state in 1s on 4G networks.
* The system should be able to work on slow networks like 2G and 3G.
  + 1. Privacy
* The gateways, sensors, and actuators should be private to the user with whom they are registered. Any other users must not be able to view and control them.
* All the personal information of the user and his devices must be private to them.
  + 1. Recoverability
* The system must be able to recover from the errors and unusable state in 10s.
  + 1. Reliability
* The Mobile App must update the state of Actuator nodes 99% of the time after we change them remotely.
  + 1. Scalability
* For every user, the system must allow them to add more sensors and actuators to scale their system.
  + 1. Supportability
* The system must be general, meaning that the system should be able to integrate any type of new sensor node without any problem.
  + 1. Usability
* Any layman must be able to learn to configure hardware after reading the user manual.

Chapter 4

# System Design

* 1. Introduction

This chapter will discuss in detail, the overall design of the “Smartly” system by using UML diagrams. The first section of this chapter will discuss the system using UML behavioral diagrams. It explains the detailed working of the system, and also shows how the system will deal with different events.

The next section will discuss in detail, the structure of the system using UML structural diagrams. This section will discuss in detail the overall architecture of the system and also the detailed designs of each subpart of the system.

* 1. Use Case Diagram

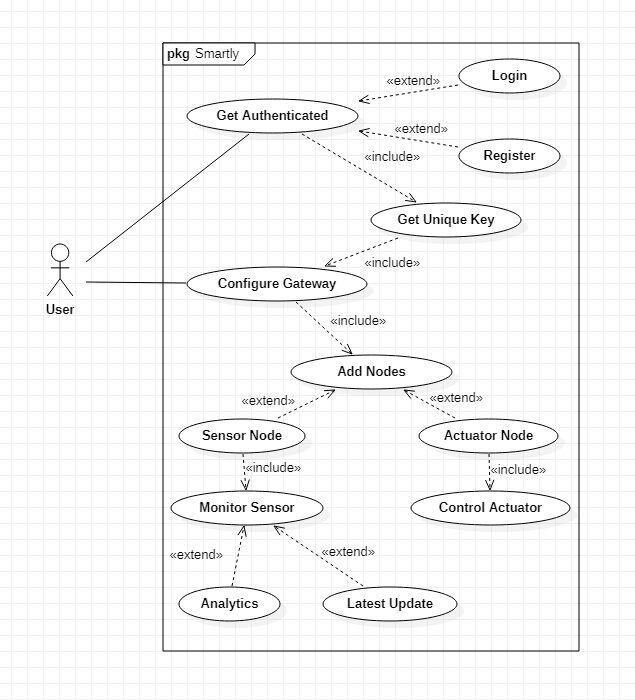


Figure 4.1 System Use Case diagram

Figure 4.1 shows the Use Case diagram of the system. The detail of each use case is given in Table 4.1.

Table 4.1 System Use Case description

|  |  |
| --- | --- |
| **Use Case** | **Description** |
| Get Authenticated | Users can create a new account and login into the system. Without the account, the user cannot access the system. |
| Get unique key | The user can get a unique API key to connect the gateway to their account. This key is unique for each user. |
| Configure gateway | Users can connect a new gateway to the account. |
| Add nodes | Users can add new sensor nodes and actuator nodes in the gateway. |
| Latest update | Users should be able to view the latest reading from the sensor nodes. |
| Analytics | Users should be able to view analytics based on the previous data from the sensor. |
| Control actuator | Users should be able to control actuator nodes. |

* + 1. Mobile App Use Case Diagram

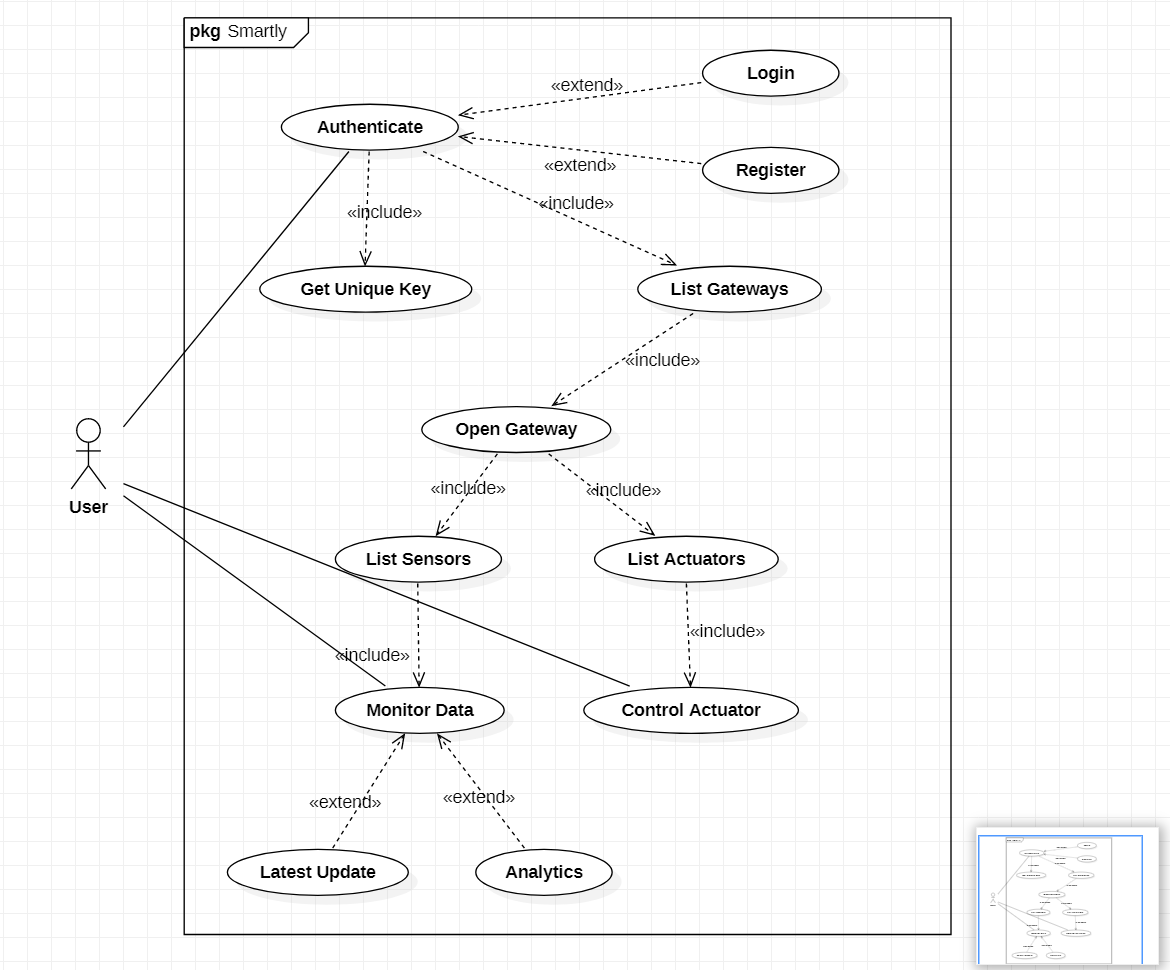


Figure 4.2 Mobile App Use Case diagram

Figure 4.2 shows the Use Case Diagram of the Mobile app. The detail of each use case is given in Table 4.2.

Table 4.2 Mobile App Use Case description

|  |  |
| --- | --- |
| **Use Case** | **Description** |
| Authenticate | Users can create a new account and login into the system from the mobile app. Without the account, the user cannot access the system. |
| Get unique key | Users can get a unique API key from the app to connect the gateway to their account. This key is unique for each user. |
| List gateway | The user can view the list of all the gateways connected to their account in the app. |
| Open gateway | Users can open any gateway from the app to view the list of sensors and actuators nodes. |
| Latest update | Users should be able to view the latest reading from the sensor nodes by using the App. |
| Analytics | Users should be able to view analytics based on the previous data from the sensor. |
| Control actuator | Users should be able to control actuator nodes by using the App. |

* + 1. Gateway Use Case Diagram

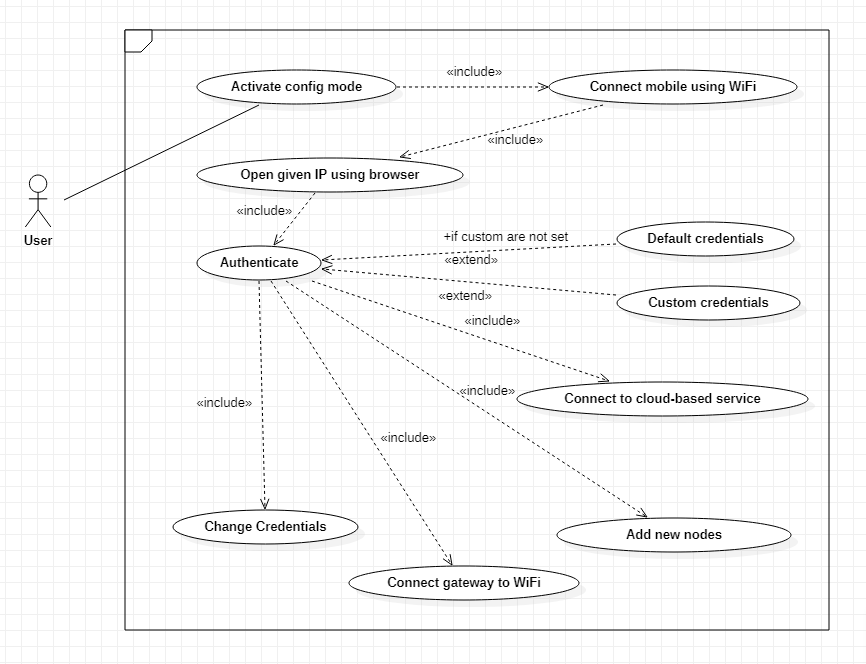


Figure 4.3 Gateway Use Case diagram

Figure 4.3 shows the Use Case Diagram of the Gateway. This diagram only shows the configuration mode of the gateway. The detail of each use case is given in Table 4.3.

Table 4.3 Mobile App Use Case description

|  |  |
| --- | --- |
| **Use Case** | **Description** |
| Activate config mode | Users can activate the config mode to configure the gateway according to their needs. |
| Connect mobile using Wi-Fi | After activating config mode user connects his/her mobile phone to the Wi-Fi Access point started by the gateway. |
| Open given IP using the browser | The user opens the default IP address of the gateway in the browser to open the configuration web app of the gateway. |
| Authenticate | Users can add credentials to the web app to access the gateway settings. |
| Connect to cloud-based service | Users can connect the gateway to their account by providing the unique key they got from Mobile App. |
| Add new nodes | Users can add new sensors and actuator nodes in the gateway. |
| Control actuator | Users should be able to control actuator nodes by using the Mobile App. |
| Connect gateway to the Wi-Fi | Users can connect the gateway to the local Wi-Fi network by providing network credentials. |
| Change credentials | The user should be able to change the credential of the gateway. |

* 1. Activity Diagram
     1. Mobile App Activity Diagram

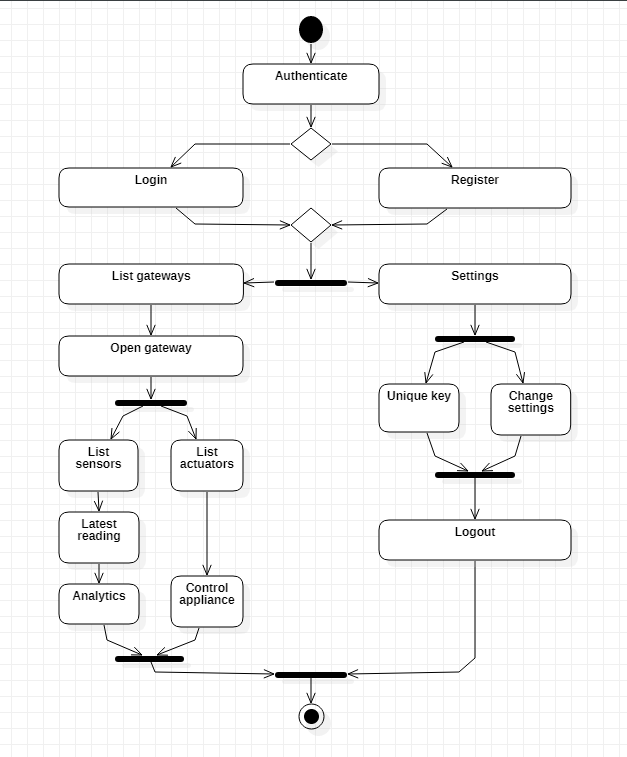


Figure 4.4 Mobile App Activity diagram

Figure 4.4 shows the Activity Diagram for the mobile app. The activity starts with the authentication of the user. In the next step, the app displays the list of gateways or settings. After the gateway, the app displays the list of sensors and the list of actuators.

* + 1. Gateway Activity Diagram

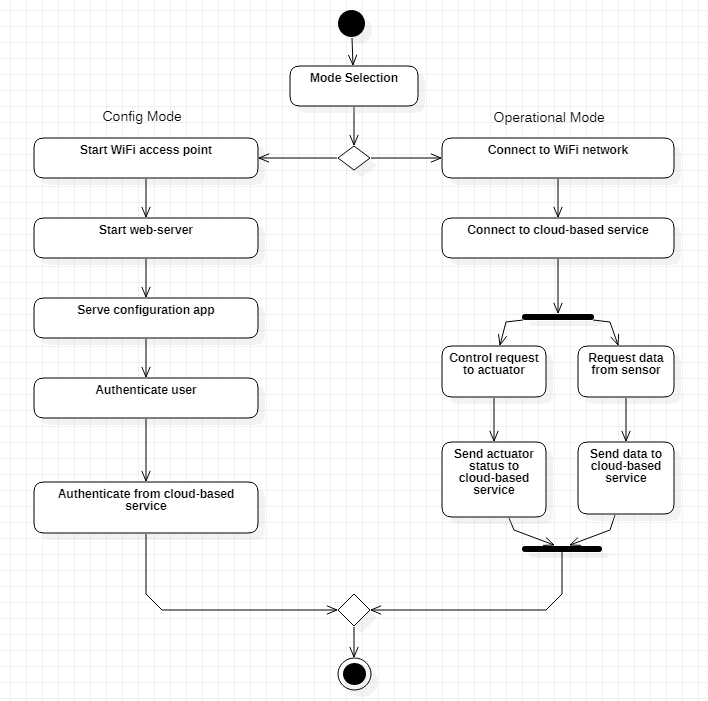


Figure 4.5 Gateway Activity diagram

Figure 4.5 shows the Activity diagram for the gateway. In config mode, activity starts with starting of the Wi-Fi Access point. Next, it starts a webserver and launches the configuration web app. After authenticating the user, new settings and nodes can be added to the gateway. In operational mode gateway connect to the Wi-Fi network and cloud-based service. Next, it sends data from sensor nodes to the cloud-based service and control requests from the cloud-based service to the actuator nodes.

* + 1. Sensor Node Activity Diagram

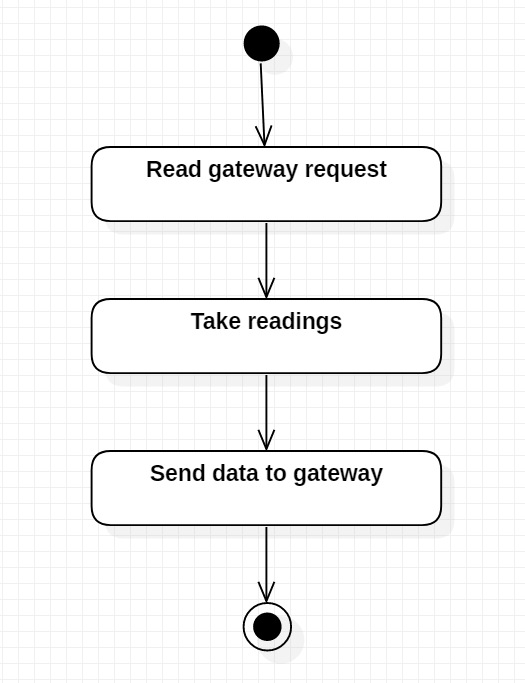


Figure 4.6 Sensor Node Activity diagram

* + 1. Actuator Node Activity Diagram

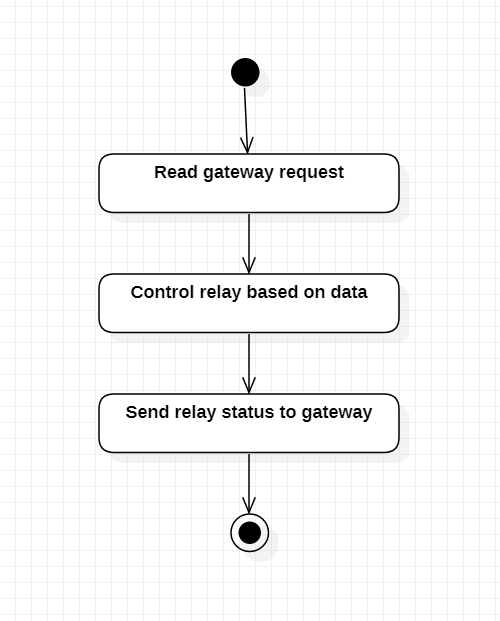
****

Figure 4.7 Actuator Node Activity diagram

* 1. Sequence Diagram

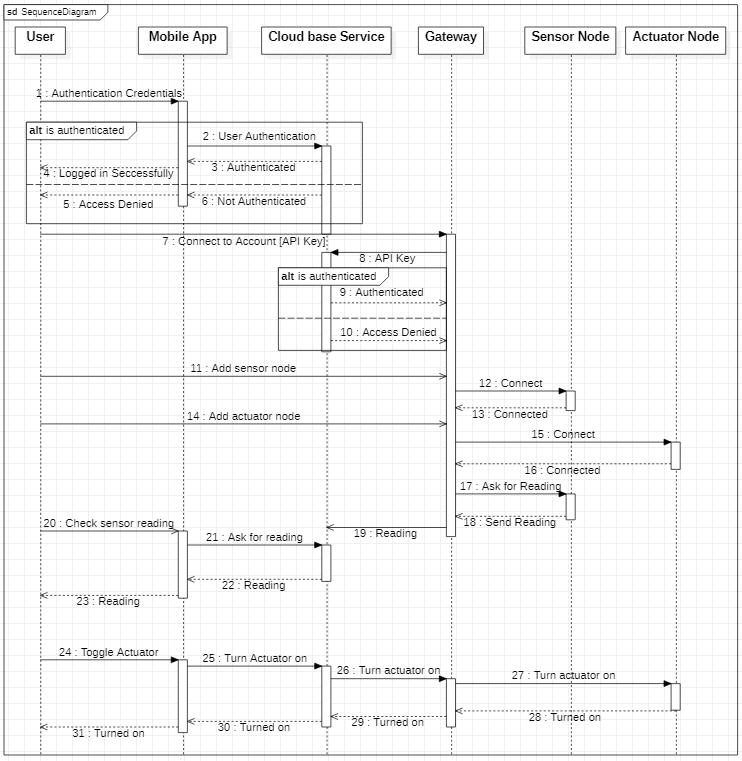


Figure 4.8 System Sequence diagram

Figure 4.8 shows the Sequence Diagram of the system. Details of the different types of messages sent by each component in the system are explained in this diagram.

* 1. Component Diagram

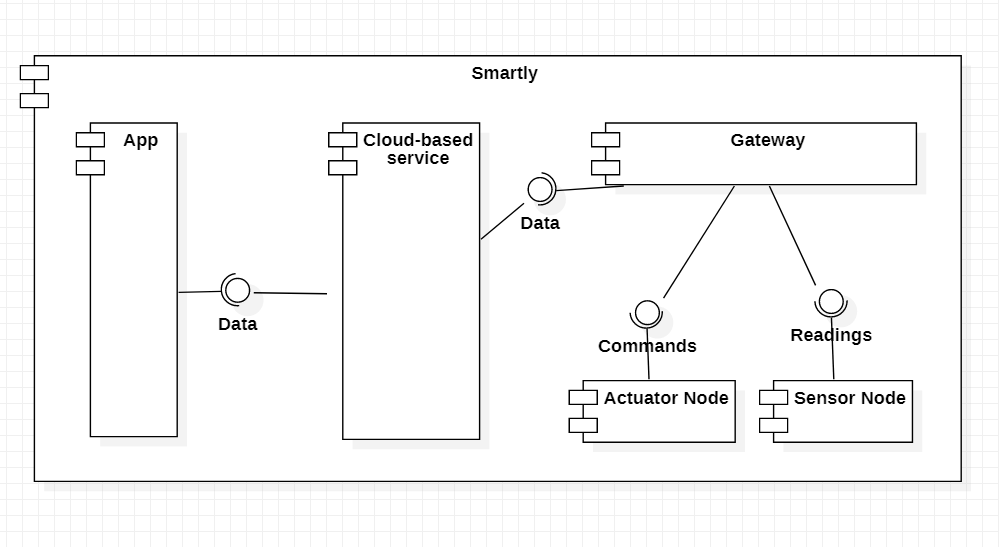
****

Figure 4.9 System Component diagram

Figure 4.9 shows the component diagram of the “Smartly” system. The system is divided into five main components. App provides the user interface for controlling and monitoring. Cloud-based service will connect IoT hardware with the App and it also stores data from sensors. Gateway is the heart of this system, it connects different sensors and actuator nodes to the cloud-based service wirelessly. Actuator nodes are responsible for controlling the appliances. The actuator is just like a switch controlled using IoT. The sensor nodes are responsible for monitoring the environment.

* + 1. Cloud-Based Service Component Diagram

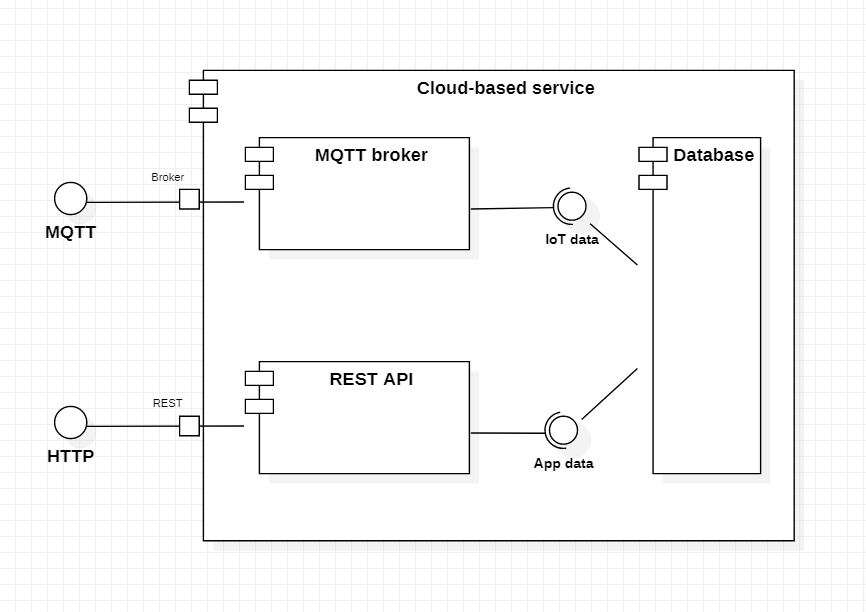


Figure 4.10 Cloud-based service Component diagram

Figure 4.10 shows the Component Diagram of the cloud-based service. It consists of three main components. MQTT broker provides the MQTT services for exchanging data in real-time with the gateway. REST API is based on the HTTP protocol and it provides services to the mobile app. The database is responsible for storing the data related to the user and IoT hardware etc.

* + 1. Gateway Component Diagram

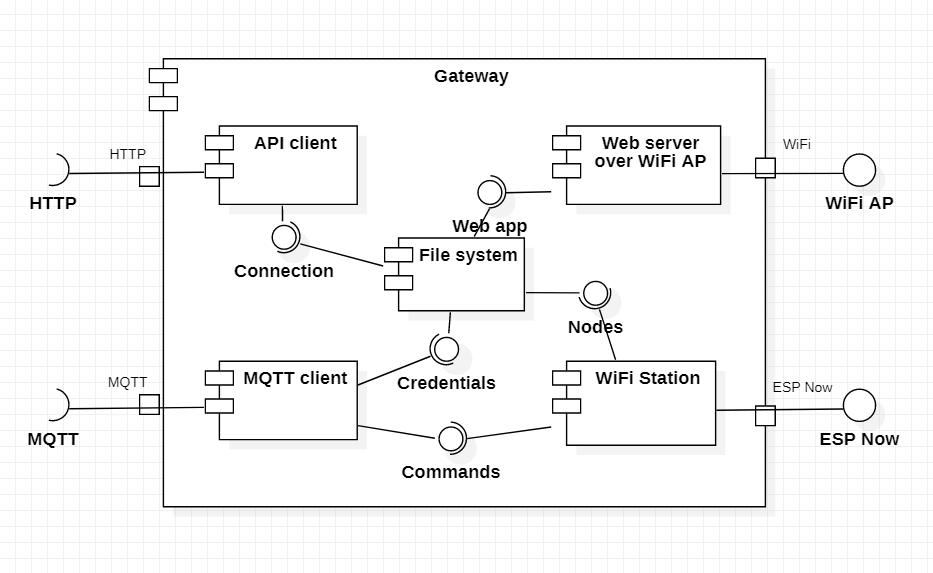


Figure 4.11 Gateway Component diagram

Figure 4.11 shows the component diagram of the gateway. Gateway is the heart of this system and it is mainly divided into five main components. API client will communicate with the cloud-based service over the HTTP protocol. It only works when the gateway will first register itself to the cloud-based service. MQTT client is responsible for communicating with cloud-based service over the MQTT protocol and it works all the time after successfully registering to cloud-based service. The file system will store the local information of the gateway, it includes nodes information and configuration web app. The webserver is responsible for serving the configuration web app. It will work over the Wi-Fi Access point. The Wi-Fi station is responsible for connecting with the sensor and actuator nodes. It uses the ESP-Now protocol for exchanging the data with the sensor and actuator nodes.

* + 1. Sensor Node Component Diagram

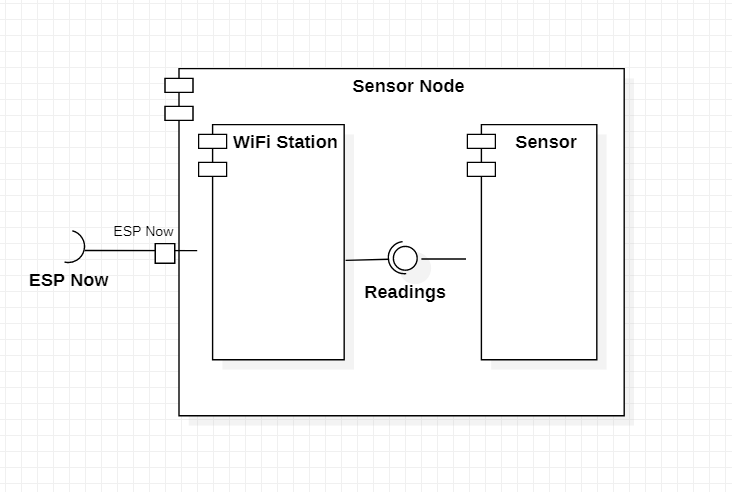
****

Figure 4.12 Sensor Node Component diagram

Figure 4.12 shows the Component Diagram of the sensor node. Wi-Fi Station is responsible for connecting with the gateway using the ESP-Now protocol. The sensor is responsible for monitoring the environment.

* + 1. Actuator Node Component Diagram

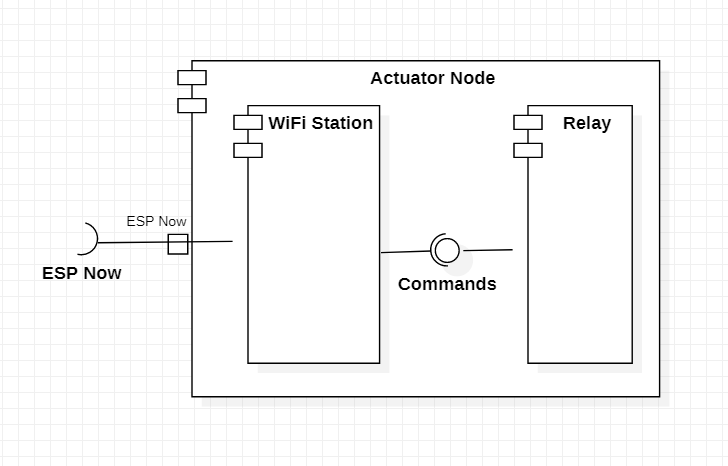


Figure 4.13 Actuator Node Component diagram

Figure 4.13 shows the Component Diagram of the actuator node. Wi-Fi Station is responsible for connecting with the gateway using the ESP-Now protocol. Relay is responsible for controlling the appliances.

* 1. Deployment Diagram

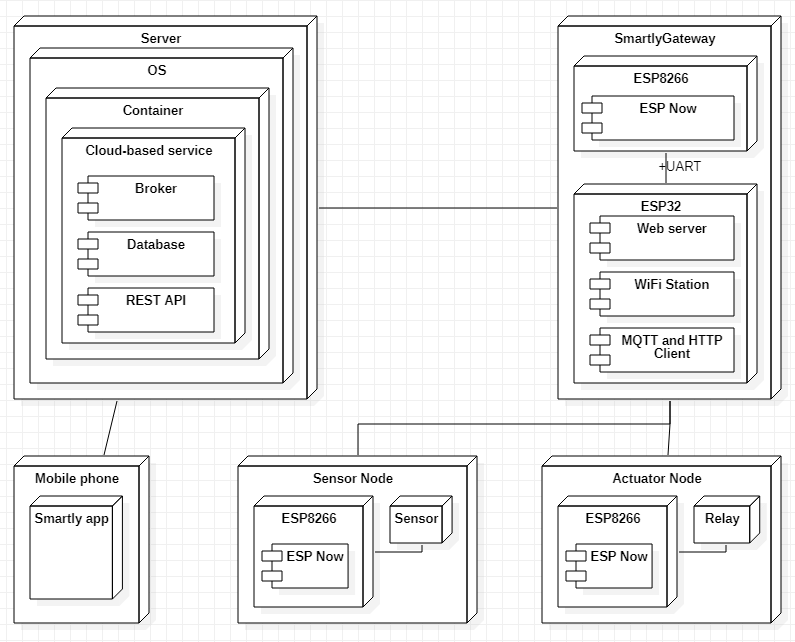


Figure 4.14 System Deployment diagram

Figure 4.14 shows the Deployment diagram for the system. The system is deployed in the distributed environment. Cloud-based service is deployed to the server and it will run inside a Docker container. The reason behind using the Docker container is to enhance the portability, performance, and speed of the cloud-based service. The mobile app will be cross-platform and sport both Android and iOS. It will be built using React-Native. Gateway will be built using custom ESP32 and ESP8266 boards. We will build the complete gateway from the scratch. Sensor and actuator nodes will be built using ESP8266 boards. Gateway, Sensor nodes, and Actuator nodes will be deployed by the user according to their needs.

* 1. Entity Relationship Diagram

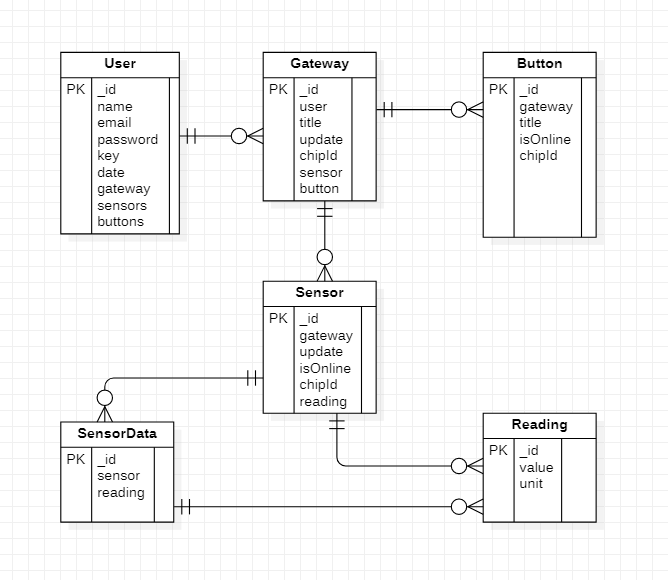
****

Figure 4.15 Entity Relationship Diagram

Figure 4.15 shows the Entity-Relationship Diagram (ERD) of the system. We are using a No-SQL database (MongoDB). There is a total of five collections in our database. User collection is responsible for storing user information. Gateway collection will store the gateway information of a specific user. Button collection will store the information about the actuator node for a specific gateway. Sensor collection will store the information about the sensor node for a specific gateway. Sensor-Data collection will store the data coming from the specific sensor. Reading collection is a nested collection inside Sensor and Sensor-Data for storing the latest readings.

Chapter 5

# Implementation

* 1. Introduction

In this chapter, we will discuss the implementation details of the “Smartly” system.

* 1. Mobile App UI (Android)
     1. Splash Screen

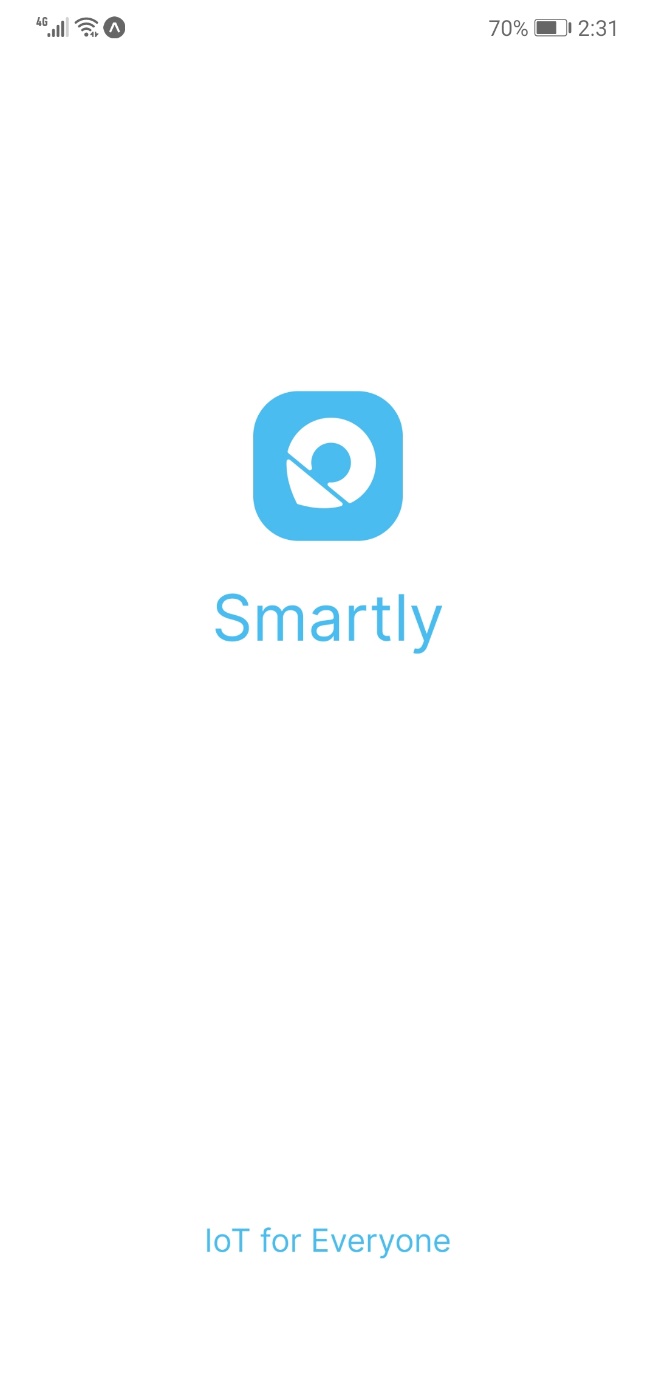


Figure 5.1 Splash Screen on Android

Splash Screen is shown when the app is launched and the App state is being prepared on the android device. After this user is either taken to Login Screen or to Home Screen.

* + 1. Login Screen

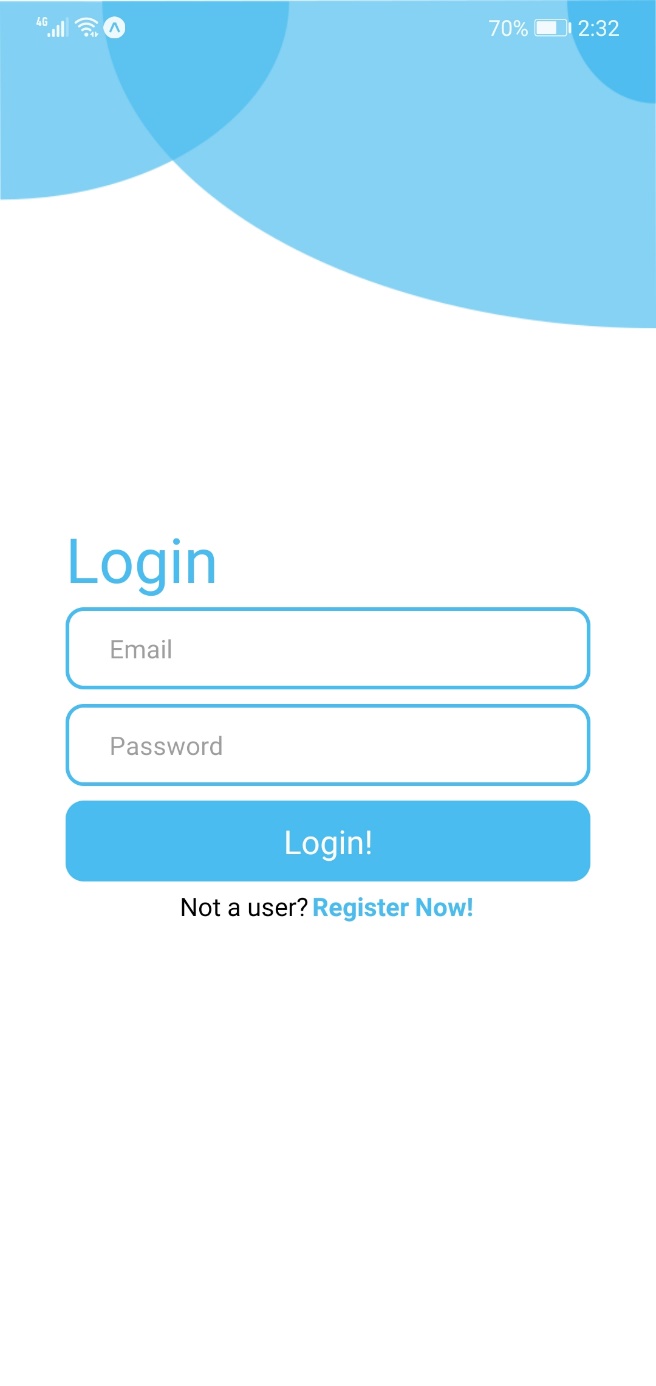


Figure 5.2 Login Screen on Android

On the login screen, the user is allowed to enter their credentials (email and password) to login into their account. This screen also handles the validation of the entered data on the client-side before sending it to the server.

* + 1. Register Screen

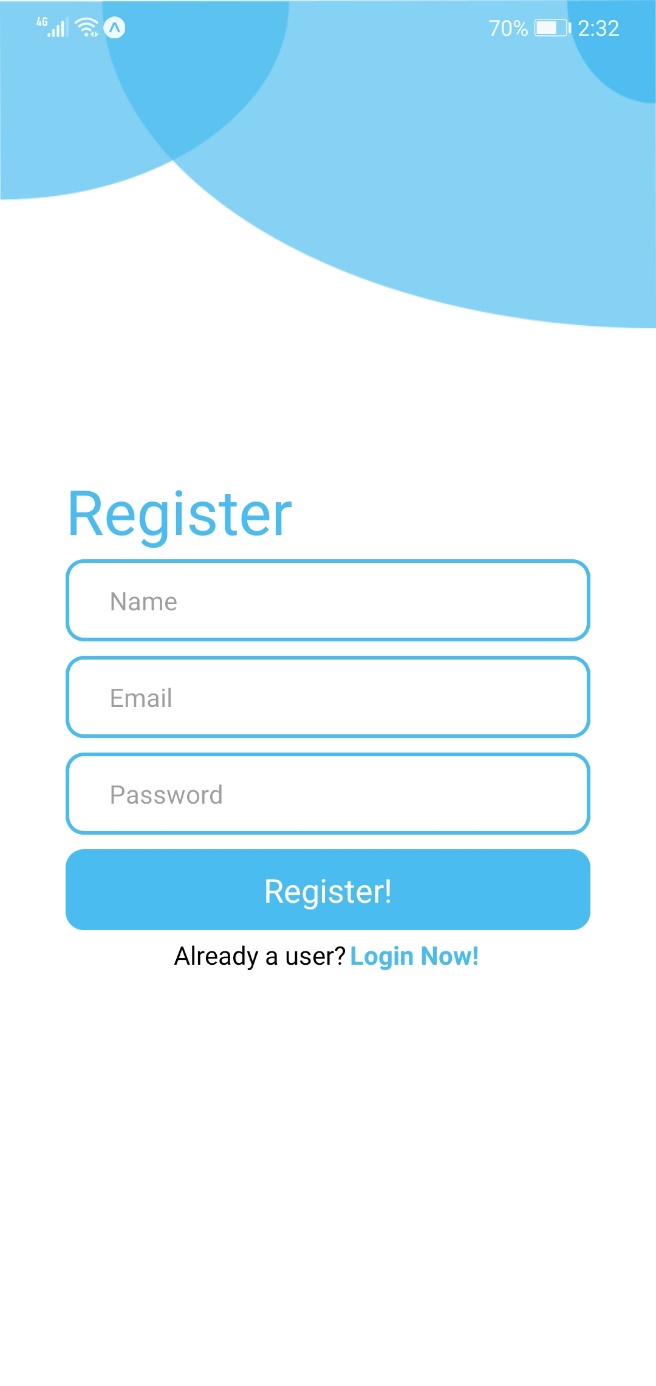


Figure 5.3 Register Screen on Android

This screen allows new users to register on our system. The user can enter their name, email, and password which gets validated on the client-side and then sent to the server, where it gets verified if registration succeeds then the user is taken to Home Screen.

* + 1. Home Screen

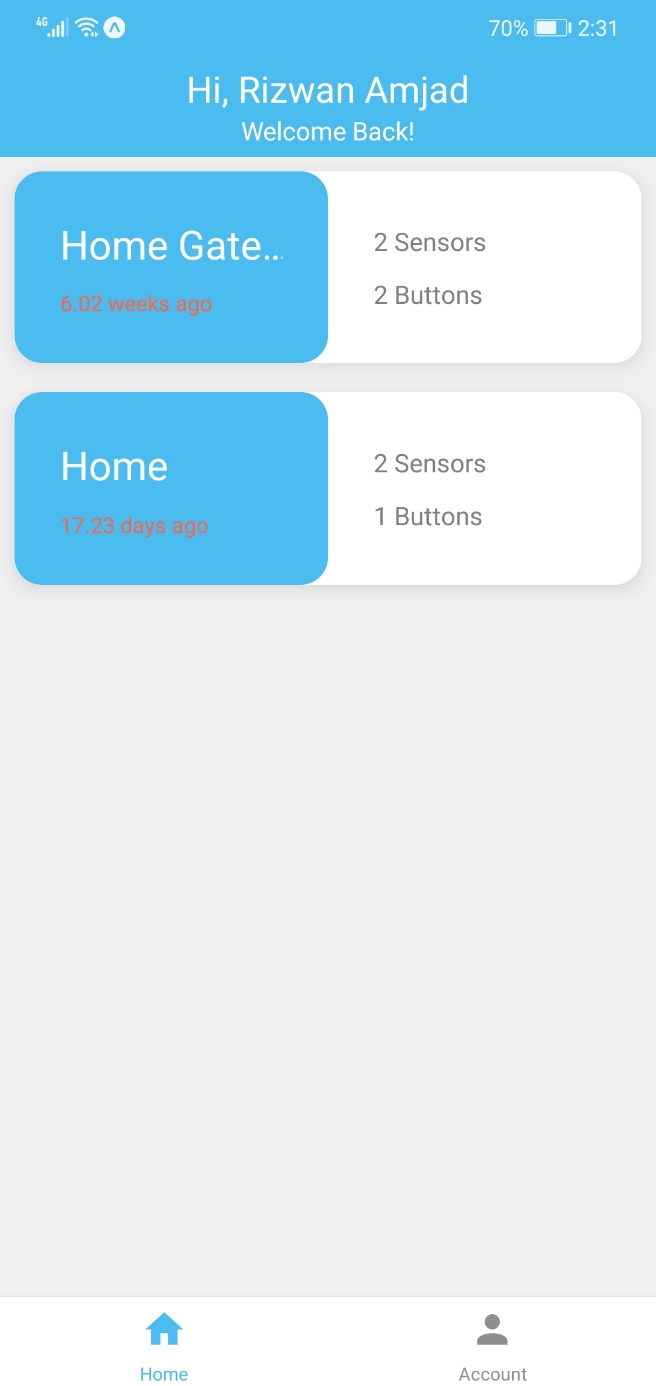


Figure 5.4 Home Screen on Android

The Home Screen lists down all the Gateways the logged-in user has registered/configured with the Smartly system. The user can also tap on Account to Navigate to the Account Screen.

Also from this screen, the user can tap on any gateway to open it, where they can view can control their sensor and actuator nodes.

* + 1. Account Screen

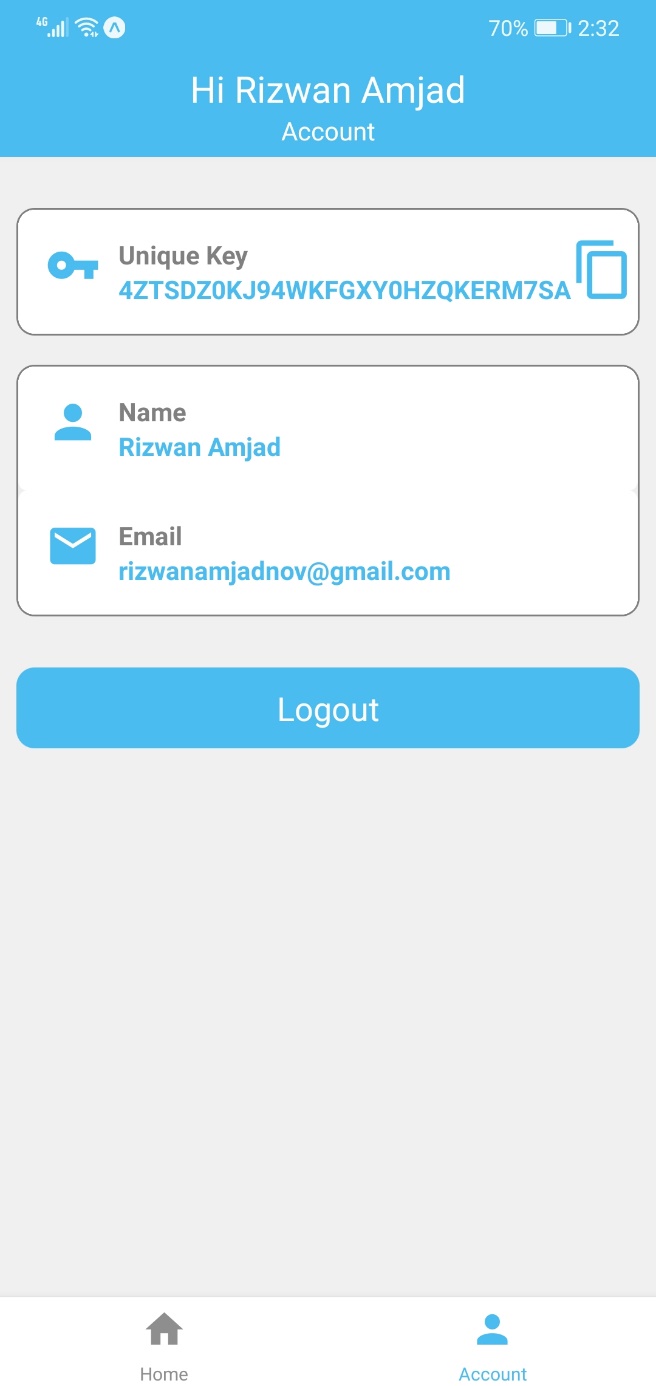


Figure 5.5 Account Screen on Android

The Account screen displays the Account information as well as the **unique key** the unique key can be copied from here by tapping on the clipboard icon.

* + 1. Sensors Screen

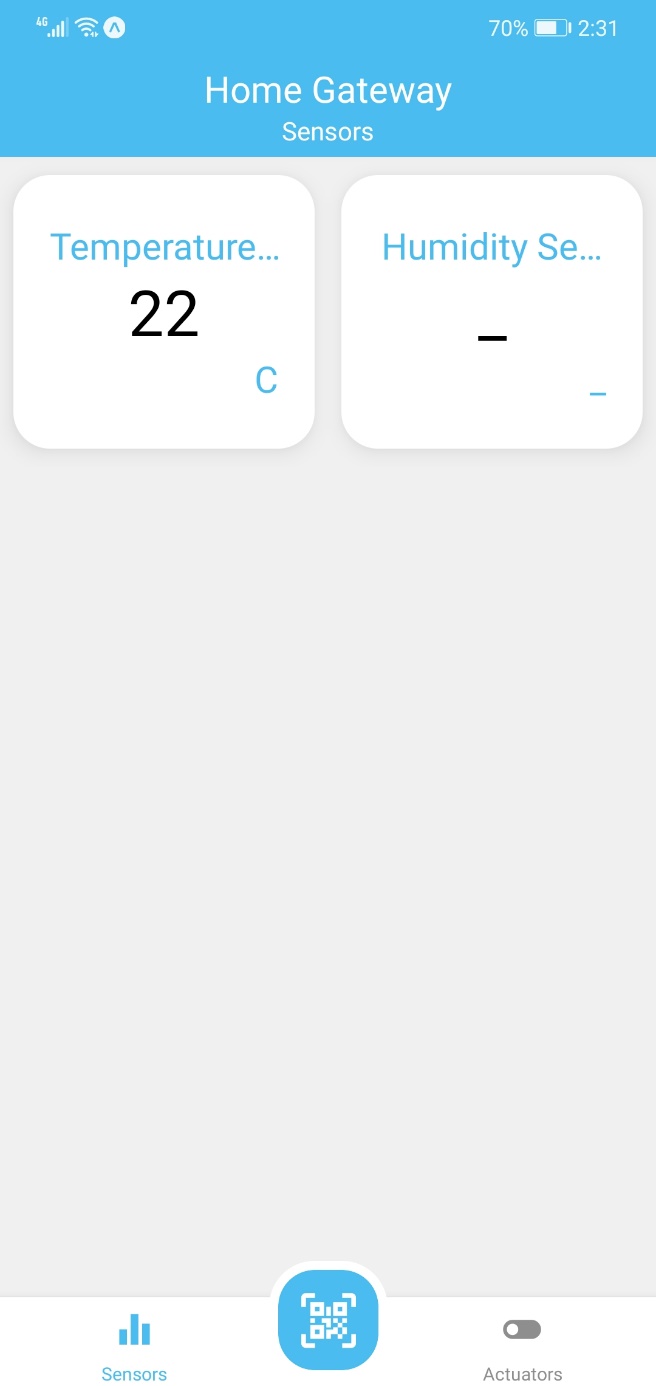


Figure 5.6 Sensor Screen on Android

This screen lists down all the sensors that are configured with a particular gateway. The user can navigate to the Actuators screen by tapping on the Actuator tab or they can configure a new sensor or an actuator by tapping on the QR icon, where they are asked to scan the QR Code.

* + 1. Actuator Screen

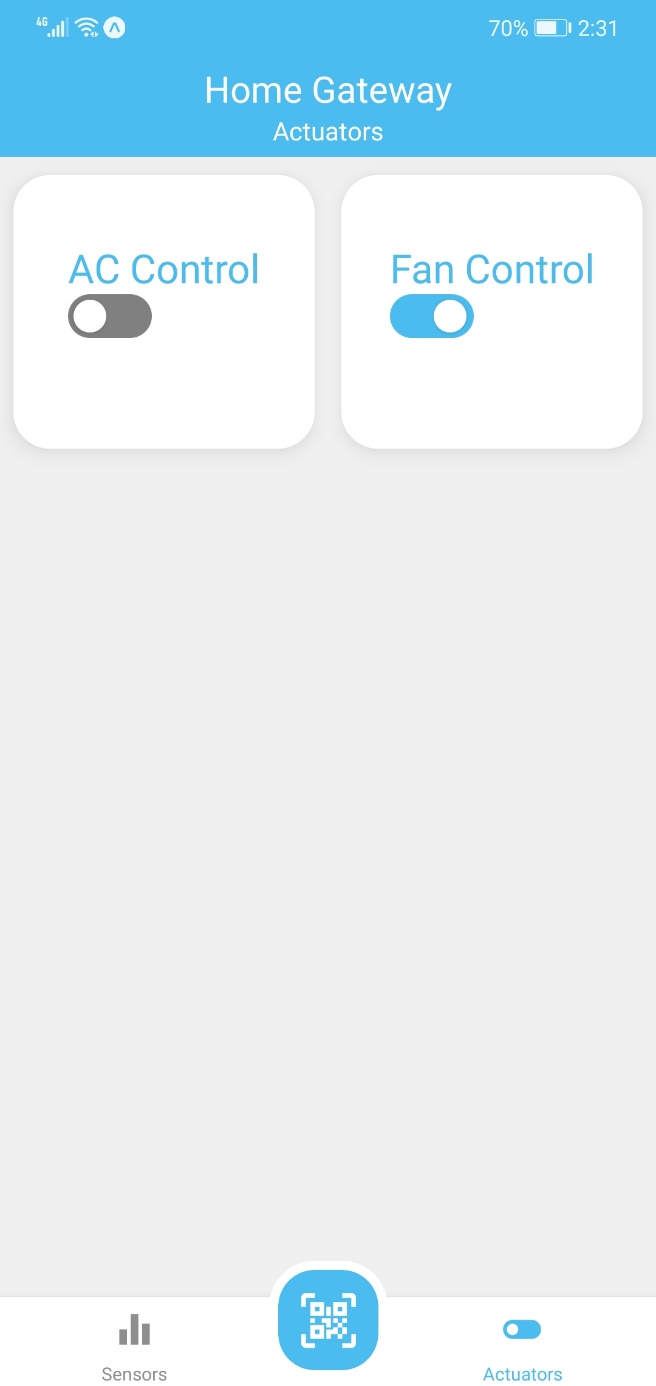


Figure 5.7 Actuator Screen on Android

Like the Sensors screen, the Actuator screen lists down all the Actuators that are registered with the Gateway. They can also control the Actuator’s state by tapping on it.

* + 1. QR-Code Scan Screen



Figure 5.8 QR-Code Scan Screen on Android

The QR Code scan screen allows the user to scan the QR Code of any node. After scanning the QR Code of the Node, the node is added to that particular gateway.

Moreover, this screen also handles if the QR Code is of the same format that our system provides.

* + 1. Analytics Screen

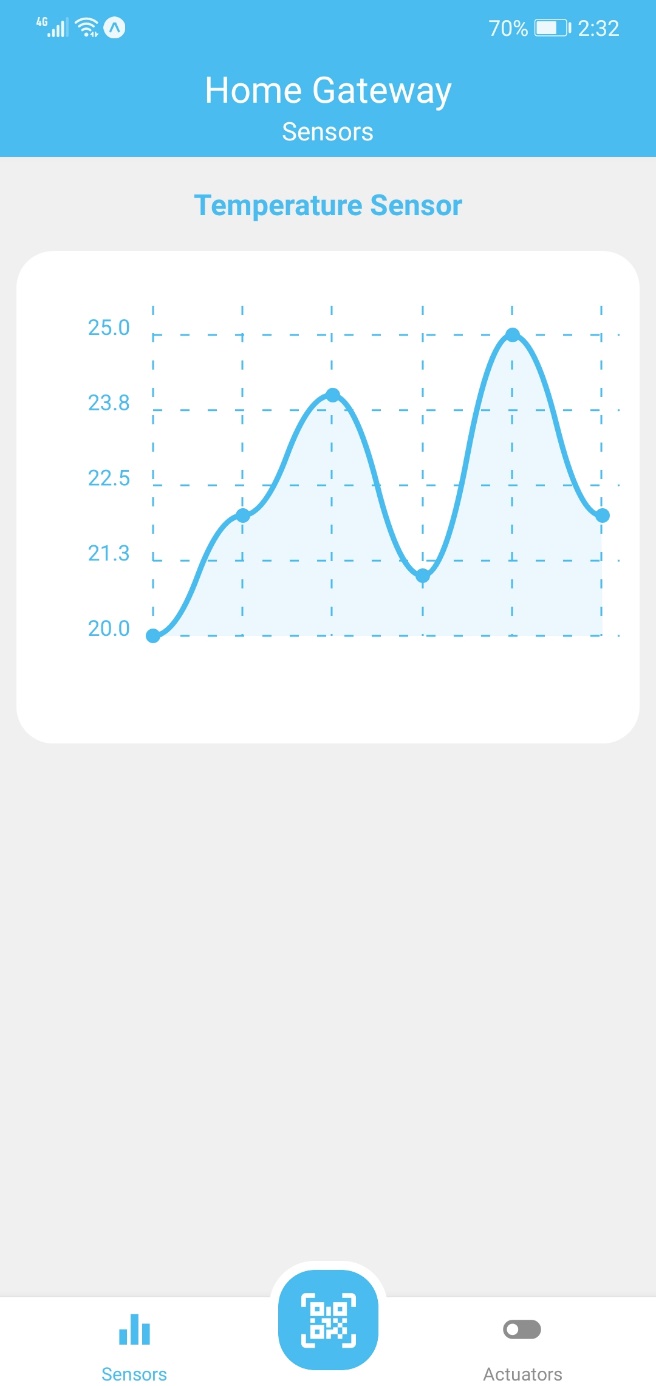


Figure 5.9 Analytic Screen on Android

The Analytics screen shows the previous sensor data in the form of a graph. The users can easily check the way sensor readings are changing graphically.

* 1. Mobile App UI (iOS)
     1. Splash Screen

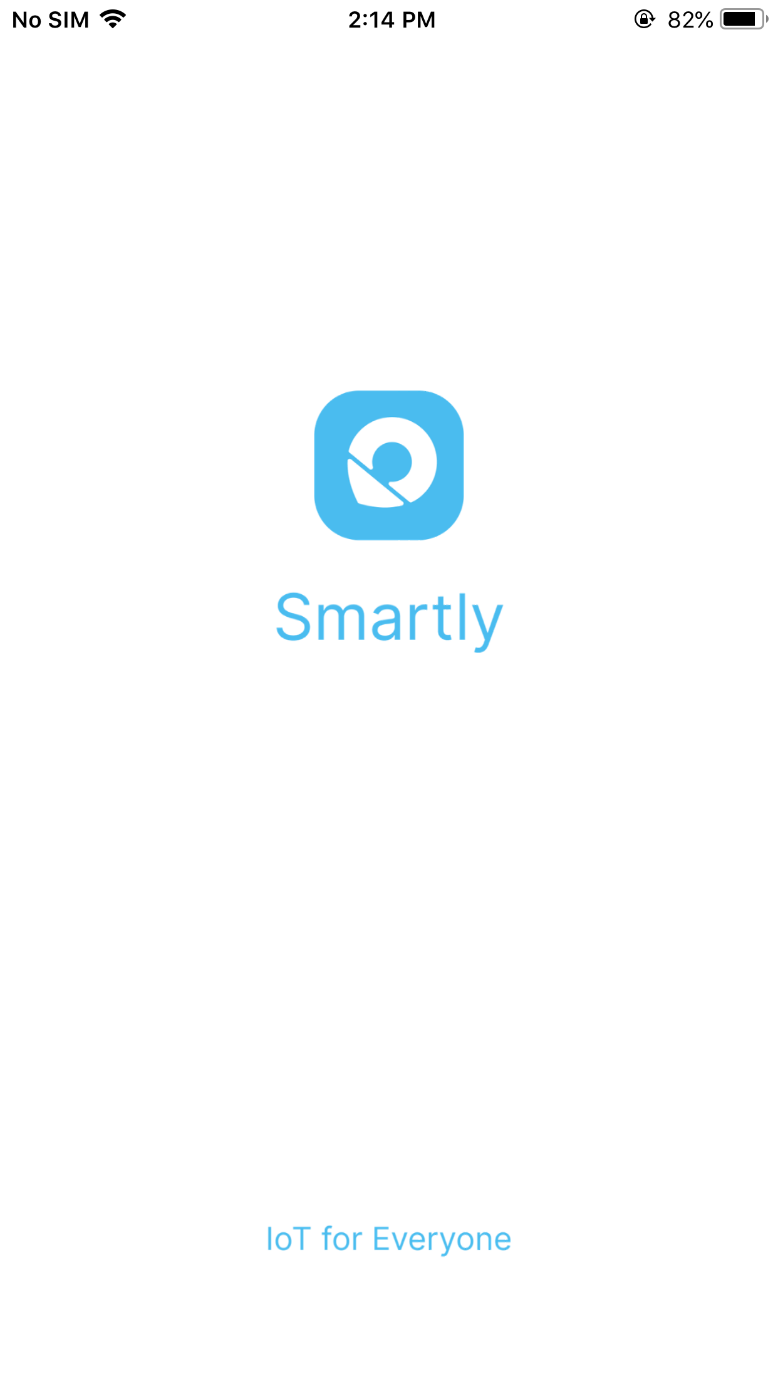


Figure 5.10 Splash Screen on iOS

Splash Screen is shown when the app is launched and the App state is being prepared on the iOS device. After this user is either taken to Login Screen or to Home Screen.

* + 1. Login Screen

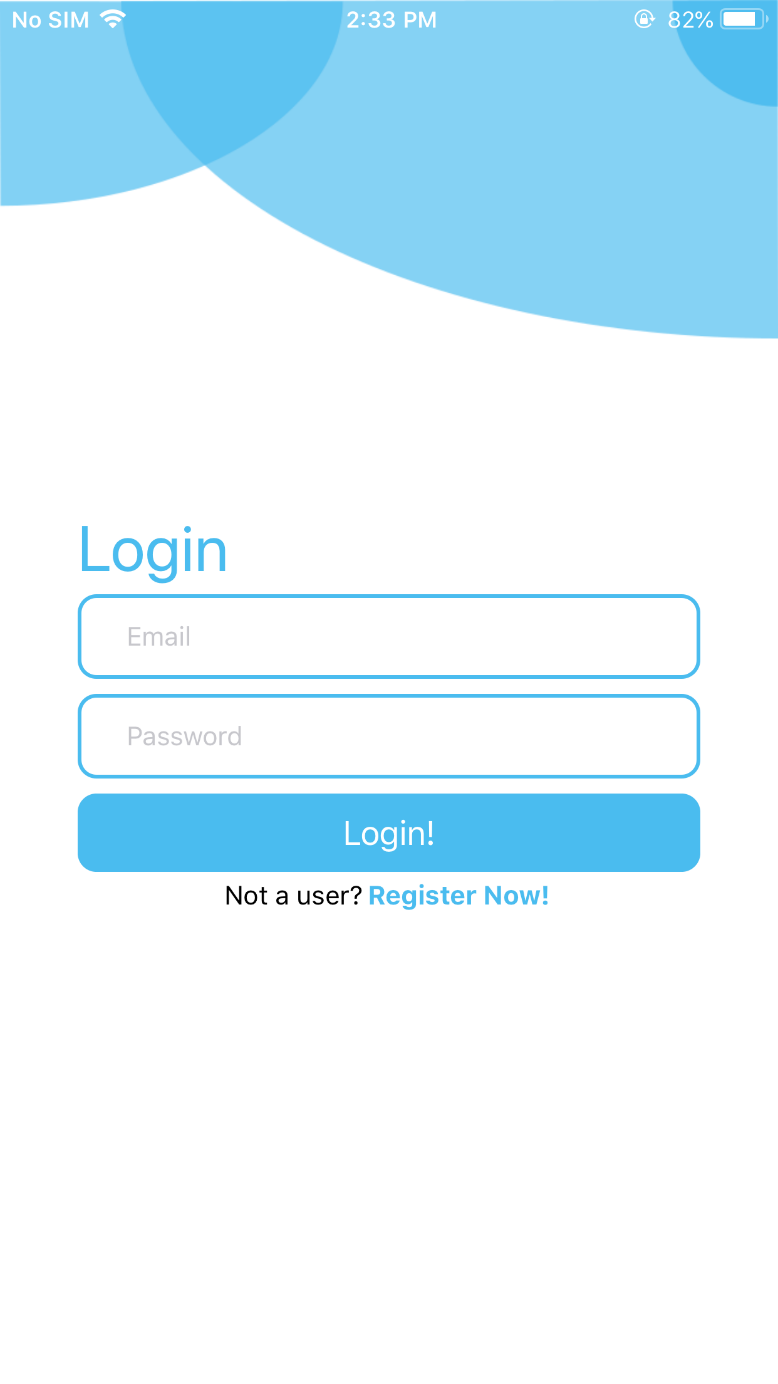


Figure 5.11 Login Screen on iOS

On the login screen, the user is allowed to enter their credentials (email and password) to login into their account. This screen also handles the validation of the entered data on the client-side before sending it to the server.

* + 1. Register Screen

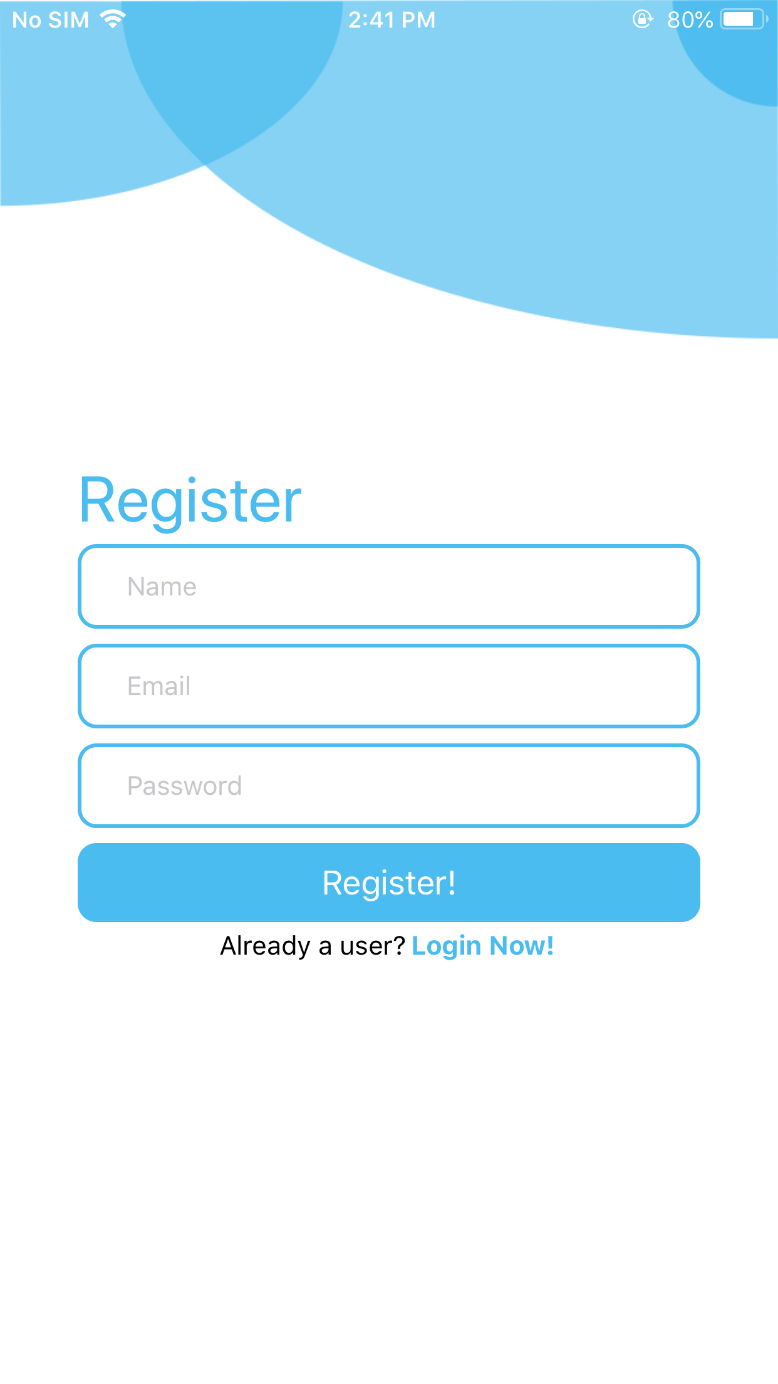


Figure 5.12 Register Screen on iOS

This screen allows new users to register on our system. The user can enter their name, email, and password which gets validated on the client-side and then sent to the server, where it gets verified if registration succeeds then the user is taken to Home Screen.

* + 1. Home Screen

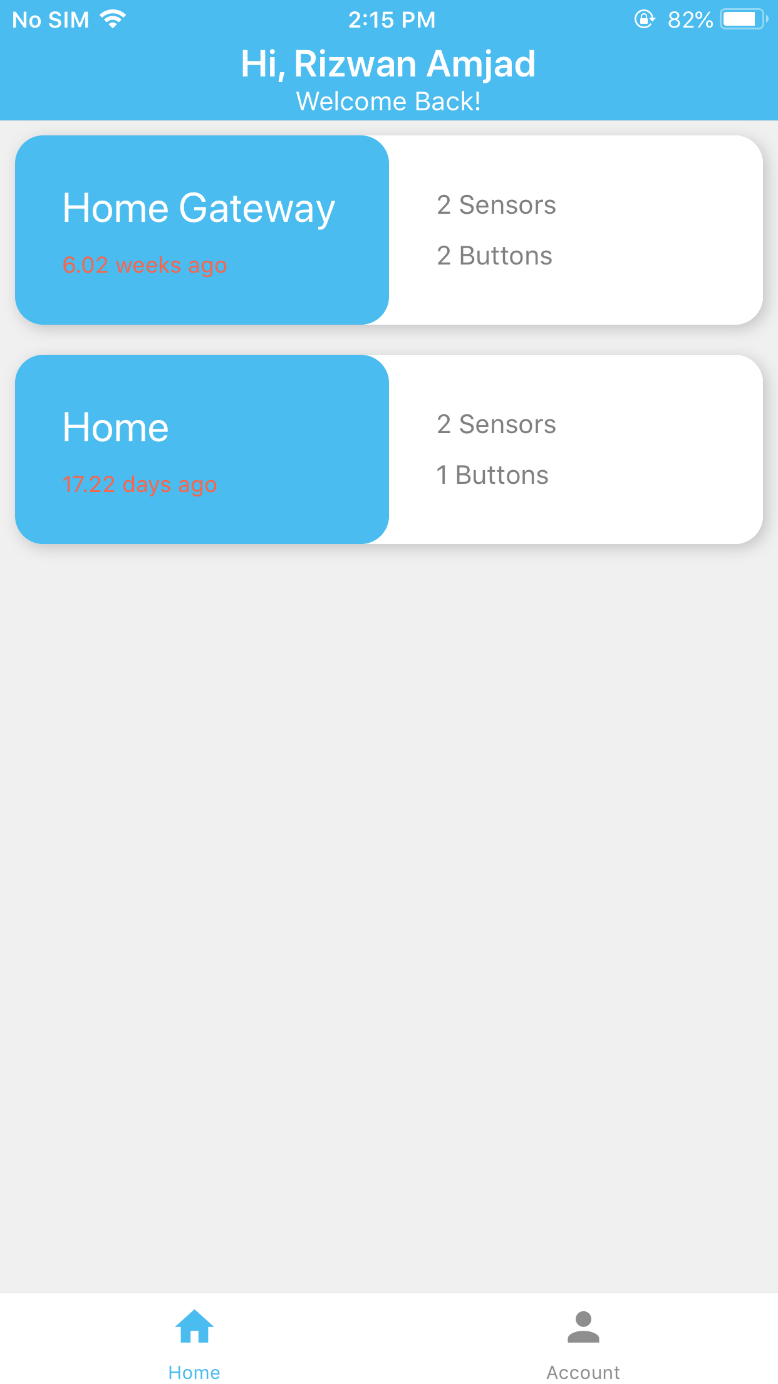


Figure 5.13 Home Screen on iOS

The Home Screen lists down all the Gateways the logged-in user has registered/configured with the Smartly system. The user can also tap on Account to Navigate to the Account Screen.

Also from this screen, the user can tap on any gateway to open it, where they can view can control their sensor and actuator nodes.

* + 1. Account Screen

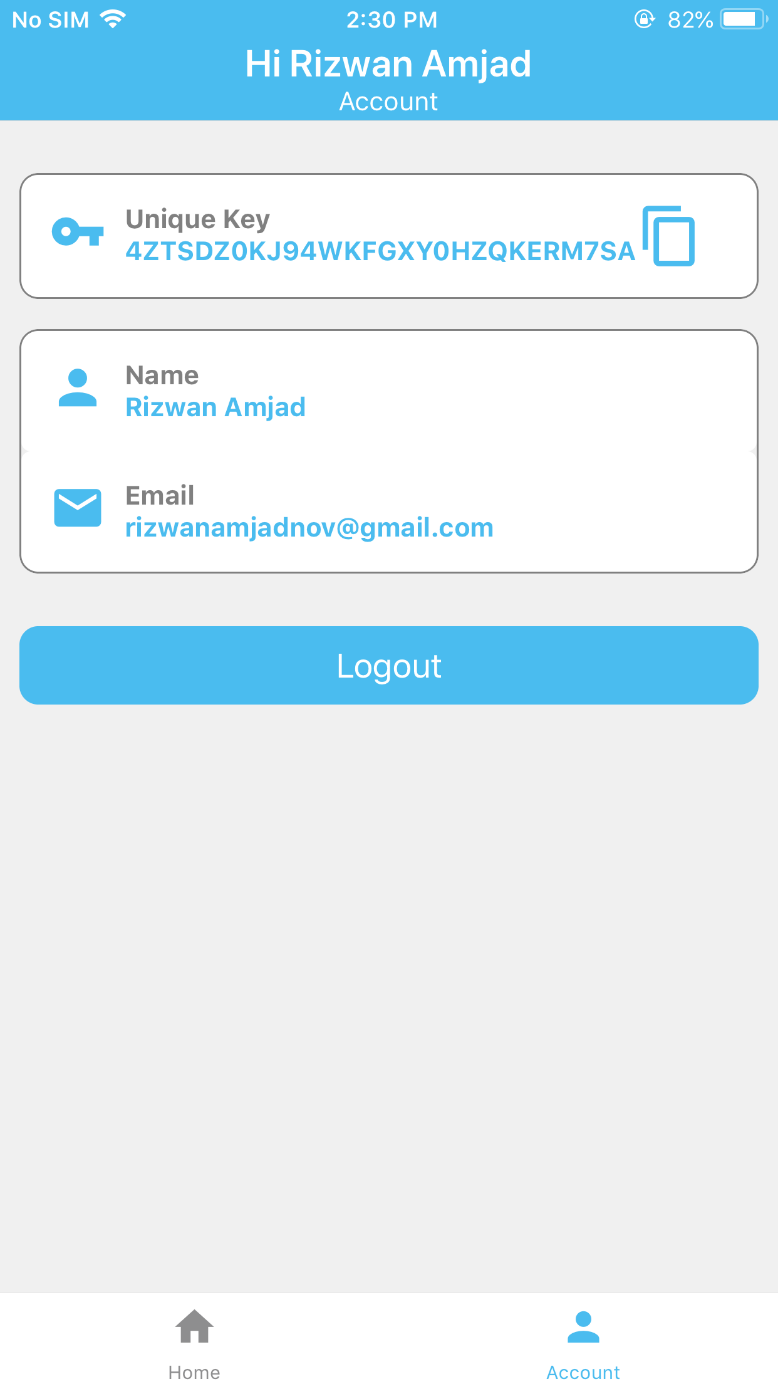


Figure 5.14 Account Screen on iOS

The Account screen displays the Account information as well as the **unique key** the unique key can be copied from here by tapping on the clipboard icon.

* + 1. Sensors Screen

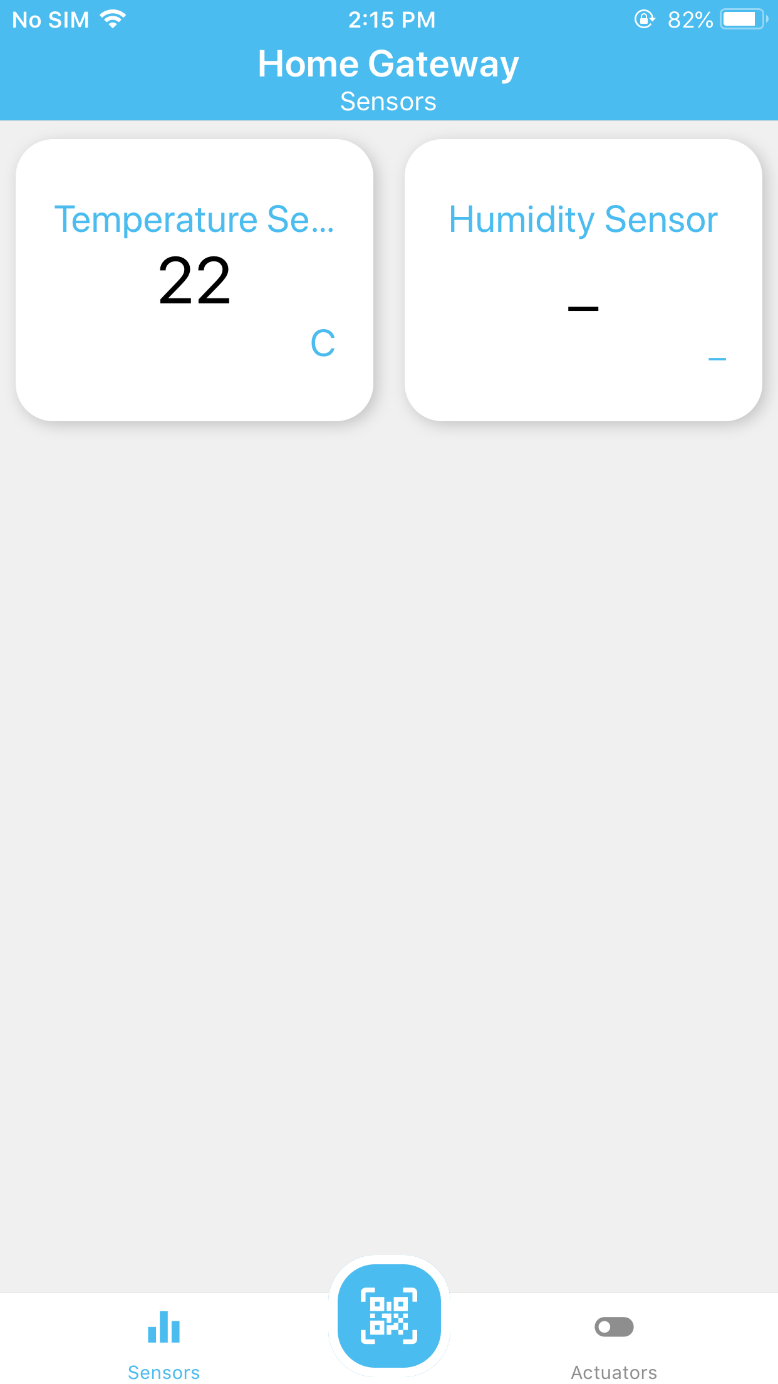


Figure 5.15 Sensor Screen on iOS

This screen lists down all the sensors that are configured with a particular gateway. The user can navigate to the Actuators screen by tapping on the Actuator tab or they can configure a new sensor or an actuator by tapping on the QR icon, where they are asked to scan the QR Code.

* + 1. Actuator Screen

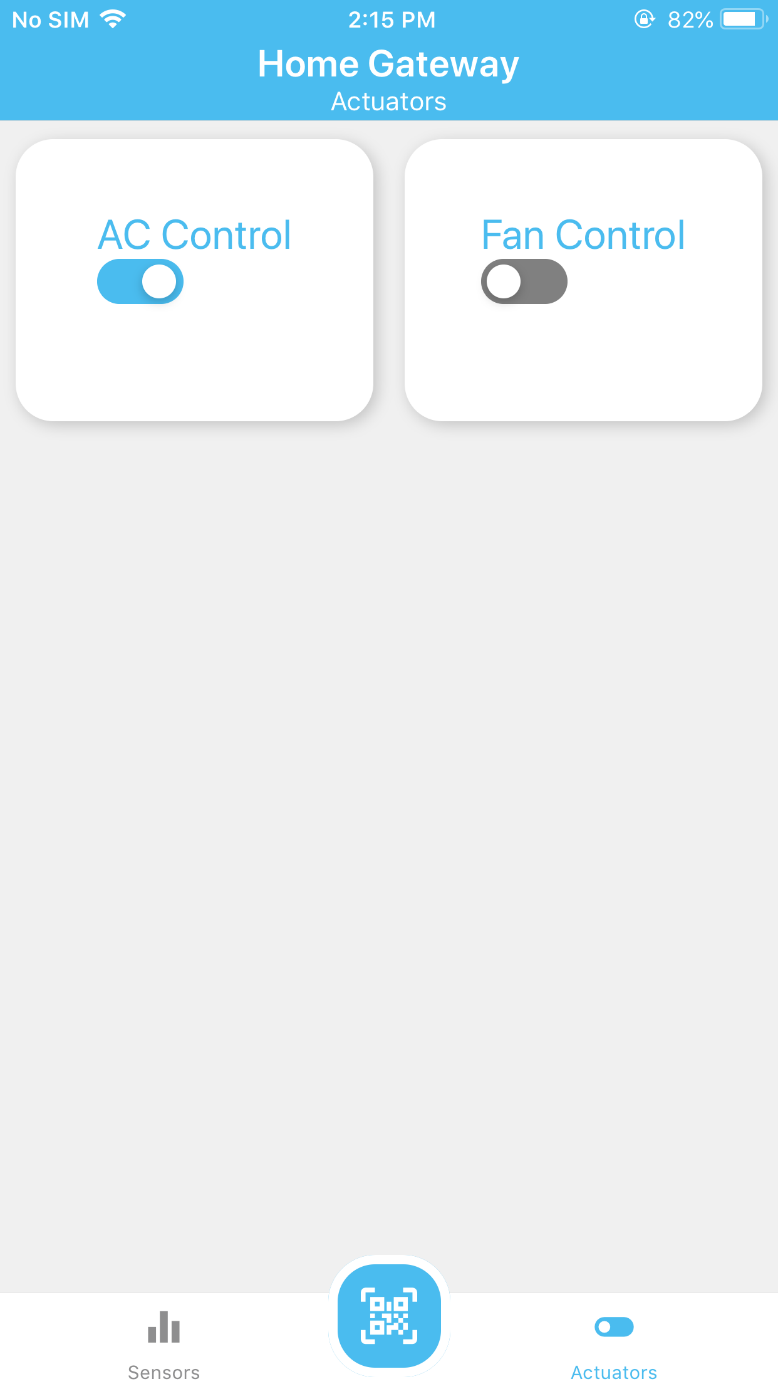


Figure 5.16 Actuator Screen on iOS

Like the Sensors screen, the Actuator screen lists down all the Actuators that are registered with the Gateway. They can also control the Actuator’s state by tapping on it.

* + 1. Qr-Code Scan Screen

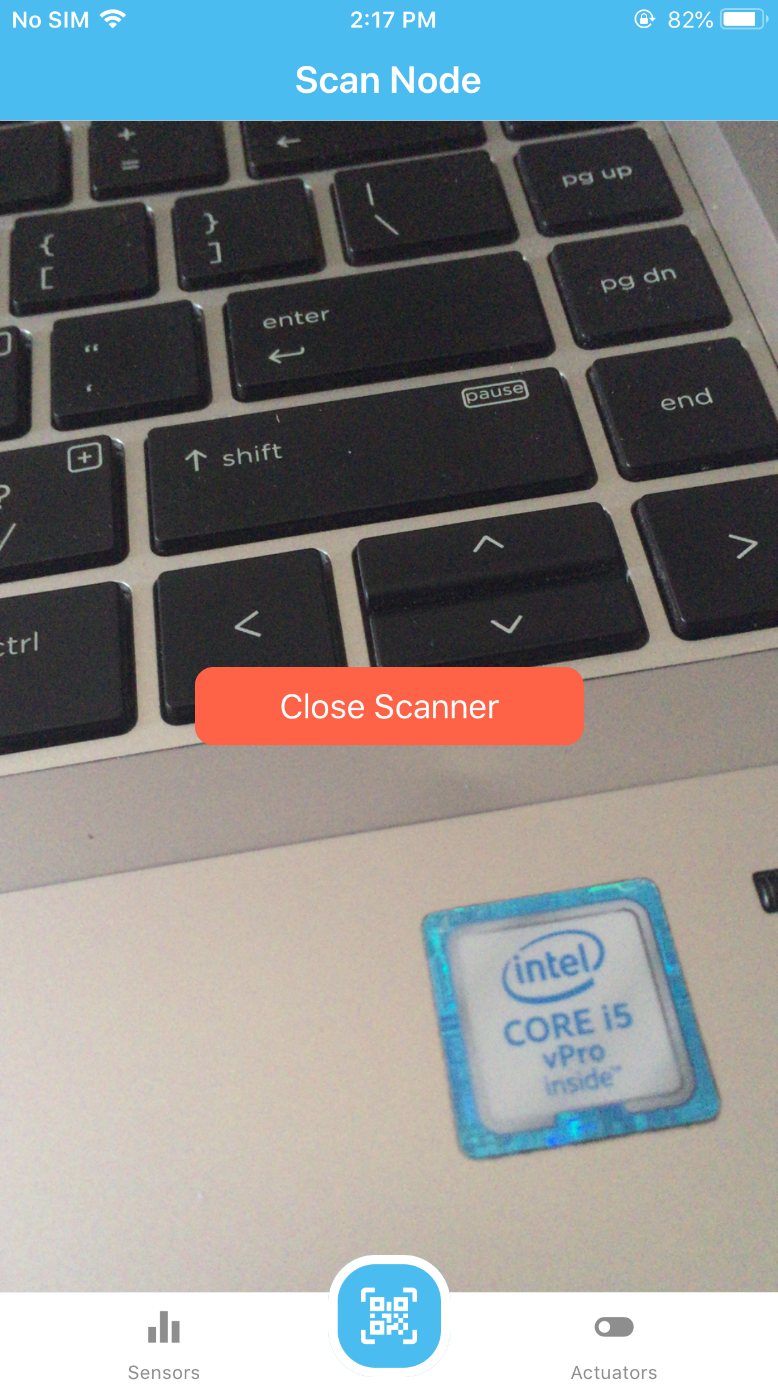


Figure 5.17 QR-Code Scan Screen on iOS

The QR Code scan screen allows the user to scan the QR Code of any node. After scanning the QR Code of the Node, the node is added to that particular gateway.

Moreover, this screen also handles if the QR Code is of the same format that our system provides.

* + 1. Analytics Screen

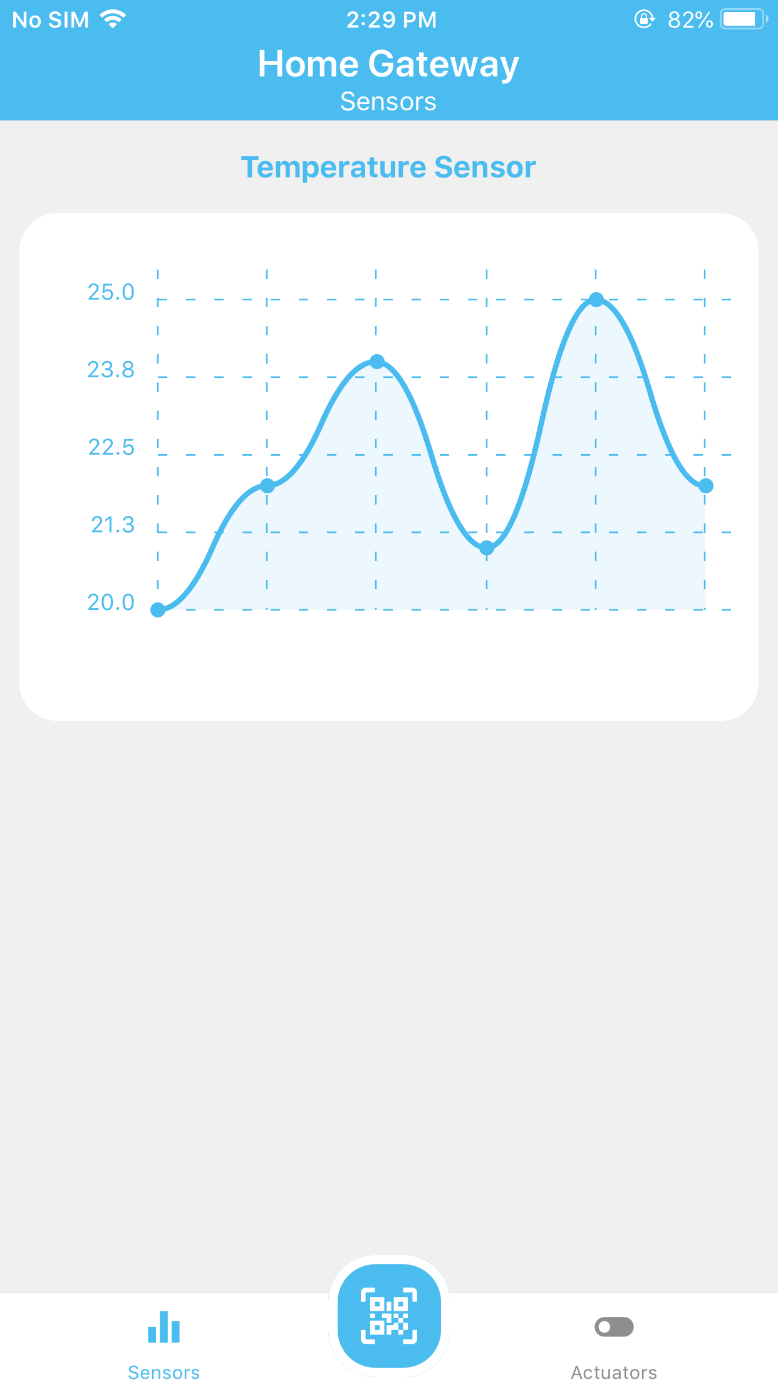


Figure 5.18 Analytic Screen on iOS

The Analytics screen shows the previous sensor data in the form of a graph. The users can easily check the way sensor readings are changing graphically.

* 1. Gateway Config Portal UI
     1. Login Screen

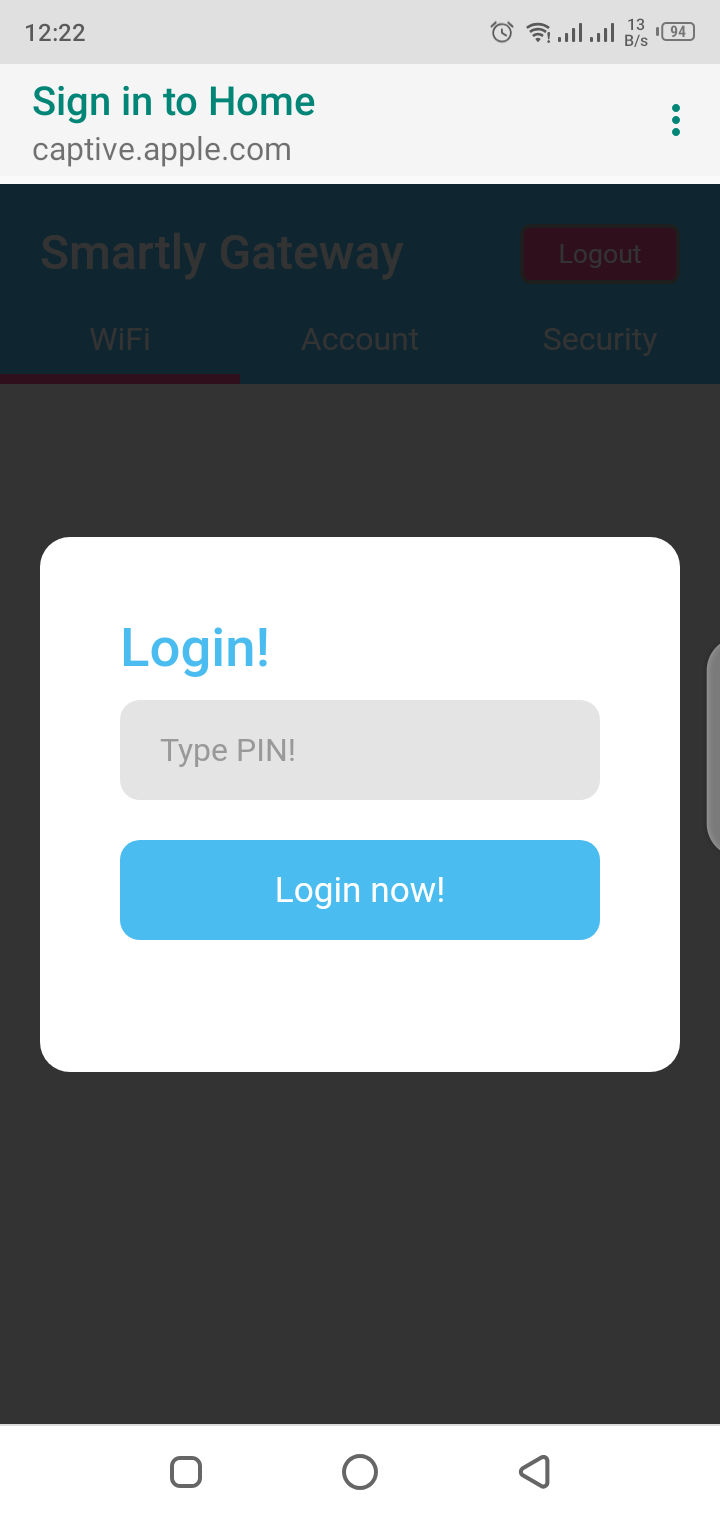


Figure 5.19 Gateway Config App Login Screen

Figure 5.19 shows the Login Screen of the gateway config portal. User can login the gateway by providing the PIN on the Login Screen.

* + 1. Wi-Fi Screen

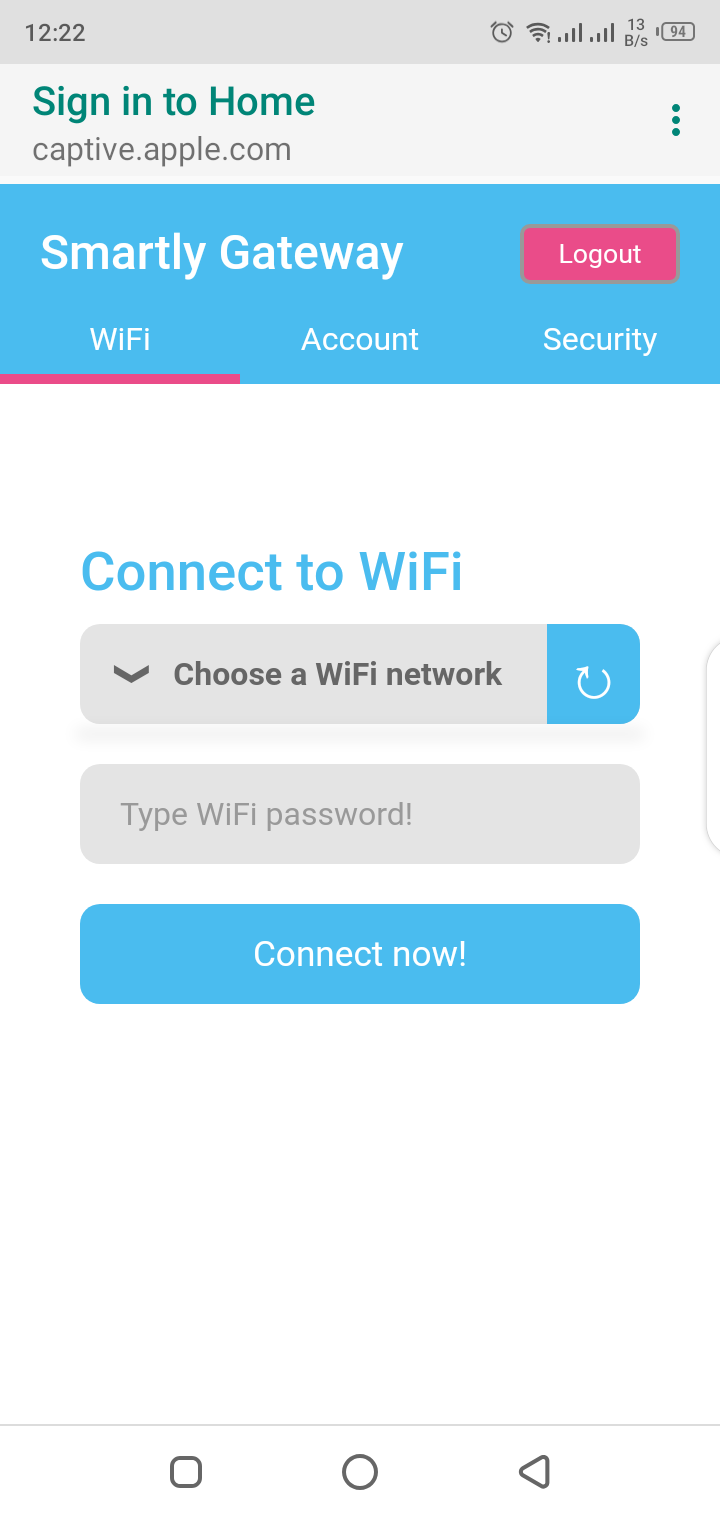


Figure 5.20 Gateway Config App Wi-Fi Screen

Figure 5.20 shows the gateway config portal Wi-Fi Screen. It displays the list of available Wi-Fi networks and user can connect the gateway to his/her network by providing the password of his Wi-Fi network.

* + 1. Account Screen

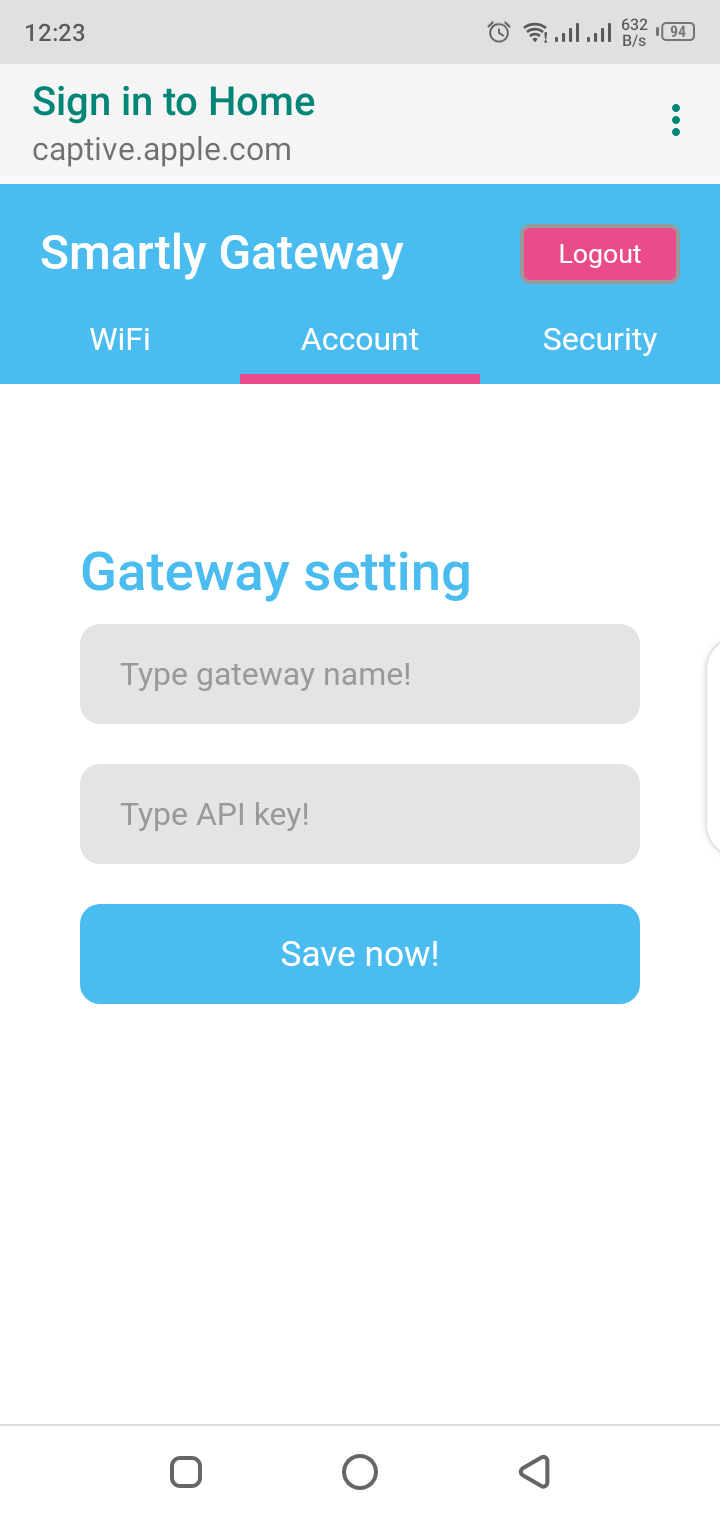


Figure 5.21 Gateway Config App Account Screen

Figure 5.21 shows the gateway config portal Account Screen. User can connect the gateway to his/her account by providing the API Key. User can also assign the name to the gateway.

* + 1. Security Screen



Figure 5.22 Gateway Config App Security Screen

Figure 5.22 shows the gateway config portal Security Screen. User can change the gateway config portal app PIN on this screen.

* 1. Gateway

The gateway is based on two different chips ESP32 and ESP-01S. ESP-32 is the **Controller** of the gateway. It is responsible all the process that deals with cloud-based service. Like HTTP, MQTT etc. It also server config portal web app for settings. It stores the setting in SPIFFS of the ESP-32 board.

ESP-01S is responsible of all the ESP-NOW operations. ESP-01S is the **Transceiver** of the gateway. It provides the communication channel with the sensor and actuator nodes to the gateway.

Both the Controller and Transceiver will communicate with each other over the UART.

* 1. Sensor Nodes
     1. Temperature Sensor

Temperature Sensor node is based on the ESP-01 and DS18B20 temperature sensor. It can measure the temperature between the ranges from **-10 °C** to **+85 °C** with an accuracy of **±0.5 °C**.

* + 1. Barometric Pressure Sensor

Barometric Pressure Sensor node is based on ESP-01 and BMP-180 sensor. It can measure the atmospheric pressure between the ranges from **300 hPa** to **1100 hPa** with an accuracy down to **0.02 hPa**.

* 1. Actuator Node

The actuator node is based on ESP-01S and single relay. It can handle the maximum load of up to **250 VAC** and **10 A**.

Chapter 6

# Testing

* 1. Introduction

In this chapter, we will discuss the testing of the “Smartly” system. The first section of this chapter will discuss the different testing techniques used at the different stages of the development. The second section has different test cases.

* 1. App Testing

The cross-platform mobile app has been developed using React-Native. We have followed the test-driven development technique during development. We also used the React Dev tool [6] for testing the app during development. We also have tested manually both on the iOS and Android phones to test out cross-platform compatibility.

* 1. Cloud-Based Service Testing

The cloud-based service has been developed using NodeJS and MongoDB. We have followed the test-driven development technique during development. We have tested the cloud-based service using tools like MQTTLens, Postman, and MongoDB Compass.

* 1. Hardware Testing

The hardware includes Gateway, Sensor nodes, and actuator nodes. They are mainly based on ESP32 and ESP8266 boards. To test them all we have followed the different approaches to different development stages. We have used test-driven development technique during the development of firmware. We have used Serial Monitor for this purpose.

For the testing of PCB circuits and hardware parts, we have used Multi-meter.

* 1. Integration Testing

Our system is network intensive and due to this reason, we performed all the integration testing manually. Firstly we have integrated Sensor and Actuator nodes to the Gateway and tested them manually. Then we connected the Gateway to the cloud-based service and test all the functionality. In the end, we connect our mobile app to the cloud-based service.

* 1. System Testing

After performing integration testing we started the system testing. We have also performed the system testing manually.

* 1. Test Cases

These are some of the important test cases we have used to test the system.

* + 1. Mobile App: Register

Table 6.1 Test cases for Register Screen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Test Condition** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Click on the Register button without entering either Name, Email, or Password | Registration should not succeed and an error message should be displayed accordingly. | Registration is not succeeded and an error message is displayed accordingly. | Pass |
| 2 | Enter the Name, Email, and Password and press the Register button | Registration should be done successfully. | Registration is done successfully. | Pass |
| 3 | Register the user with an email that has already been registered | Registration should fail and an Error should be displayed saying the email is already registered. | Registration failed and an Error is displayed saying the email is already registered. | Pass |

* + 1. Mobile App: Login

Table 6.2 Test cases for Login Screen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Test Condition** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Enter the wrong credentials (email and password) and press login | Login should fail with an error message saying Invalid email or Password | Login failed with an error message saying Invalid email or Password | Pass |
| 2 | Enter valid email and password then press login | Login should Succeed | Login Succeeded | Pass |
| 3 | Press Login Button without entering the Email and Password. | Login should fail with an error message saying enter the particular information (email or password) | Login failed with an error message saying enter the particular information (email or password) | Pass |

* + 1. Mobile App: Gateways

Table 6.3 Test cases for Gateway Screen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Test Condition** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Swipe from top to down (pull to refresh) | The gateways should be refreshed from cloud-based service | The gateways get updated from cloud-based service | Pass |
| 2 | Tap on a gateway | The particular gateway should be opened with the sensors and actuators tabs on the screen. | The particular gateway is opened with the sensors and actuators tabs on the screen. | Pass |

* + 1. Mobile App: Sensors

Table 6.4 Test cases for Sensor Screen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Test Condition** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Swipe from top to down (pull to refresh) | The sensors should be refreshed from cloud-based service | The sensors get updated from cloud-based service | Pass |
| 2 | Tap on a particular sensor | The analytics screen of the particular sensor should be opened. | The analytics screen of the particular sensor is opened | Pass |

* + 1. Mobile App: Actuators

Table 6.5 Test cases for Actuator Screen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Test Condition** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Swipe from top to down (pull to refresh) | The actuators should be refreshed from cloud-based service | The actuators get updated from cloud-based service | Pass |
| 2 | Tap on a particular actuator’s toggle button | The particular actuator’s state should be toggled | The particular actuator’s state is toggled | Pass |

* + 1. Mobile App: QR Code Scan

Table 6.6 Test cases for QR Code Scanner Screen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Test Condition** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Scan sensor node’s QR code, from a particular gateway | The new sensor node should be added to the particular Gateway. And open sensors screen | The new sensor node is added to the particular Gateway. And opens the sensors screen | Pass |
| 2 | Scan actuator node’s QR code, from a particular gateway | The new actuator node should be added to the particular Gateway. And open actuators screen | The new actuator node is added to the particular Gateway. And opens the actuators screen | Pass |
| 3 | Scan a random (non-standard for the smartly app) | Should show an error message for the invalid QR code format | Shows an error message for the invalid QR code format | Pass |

* + 1. Mobile App: Account screen

Table 6.7 Test cases for Account Screen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Test Condition** | **Expected Result** | **Actual Result** | **Status** |
| 1 | Tap on Gateway Key | The gateway key should be copied to the mobile’s clipboard | The gateway key is copied to the clipboard | Pass |

Chapter 7

# Conclusion and Future Work

* 1. Conclusion

We have completed all the initially proposed functionalities which include

* A cross-platform mobile app using React Native
* Cloud-based service using NodeJS and MongoDB
* IoT gateway using ESP32, ESP8266-01S and ReactJS
* Actuator node using ESP8266-01S and Relay
* Temperature Sensor node using ESP8266-01, DS18B20
* Barometric pressure sensor node using ESP8266-01, BMP180
* All the hardware part uses C++ with Arduino framework and PlatformIO

We also added some extra functionalities which were not proposed initially. Like QR-code scanning in the app to add a new sensor and actuator node to the gateway.

* 1. Future Work

This system work as a base for us. We have designed and developed this system for long-term use. Its structure can easily accommodate changes without any problem. We are planning to add the functionalities in the future.

* Shifting from HTTP to CoAP (Constrained Application Protocol) for improving system performance and speed.
* Development of a more powerful and custom MQTT broker, which can handle a large amount of IoT devices.
* Smart paring between IoT nodes and gateway.
* BLE (Bluetooth Low Energy) support for configuring the gateway.
* Shifting everything to the app including gateway configuration.
* And many more things…

Chapter 8

# References

**References**

|  |  |
| --- | --- |
| [1] | Vebuso, 15 02 2018. [Online]. Available: https://www.vebuso.com/2018/02/idc-80-billion-connected-devices-2025-generating-180-trillion-gb-data-iot-opportunities/#:~:text=IDC%20found%20that%20there%20will,the%20data%20created%20in%202020.. [Accessed 05 06 2022]. |
| [2] | weforum, "4th Industrial Revolution," 14 01 2016. [Online]. Available: https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/. |
| [3] | 1 10 2021. [Online]. Available: https://blynk.io/. |
| [4] | G. C. I. Core, 01 06 2022. [Online]. Available: https://cloud.google.com/solutions/iot. |
| [5] | ThingSpeak, 01 06 2022. [Online]. Available: https://thingspeak.com/. |
| [6] | Facebook, "React Developer Tools," [Online]. Available: https://www.npmjs.com/package/react-devtools. |