

# **LOW COST HEALTH MONITORING SYSTEM**

**A Project Report**

Submitted in partial fulfilment of the  
requirements for the award of the Degree of

**BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)**

**By**

**Ritik Rajesh Samanta**

Seat Number: \_\_\_\_\_

**Under the esteemed guidance of**

**Mrs. Prachi Mahajan**

**Assistant Professor, Department of Information Technology**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**VIDYALANKAR SCHOOL OF INFORMATION TECHNOLOGY**

**(Affiliated to University of Mumbai)**

**MUMBAI, 400 037**

**MAHARASHTRA**

**2019 - 2020**

# VIDYALANKAR SCHOOL OF INFORMATION TECHNOLOGY

(Affiliated to University of Mumbai)

MUMBAI-MAHARASHTRA-400037

## DEPARTMENT OF INFORMATION TECHNOLOGY



### CERTIFICATE

This is to certify that the project entitled, "**Low Cost Health Monitoring System**", is bonafied work of **Ritik Rajesh Samanta** bearing Seat No: \_\_\_\_\_ submitted in partial fulfilment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

**Internal Guide**

**Coordinator**

**Internal Examiner**

**External Examiner**

**Date:**

**College Seal**

**Principal**

**VIDYALANKAR SCHOOL OF INFORMATION TECHNOLOGY**

(Affiliated to University of Mumbai)


MUMBAI-MAHARASHTRA-400037

**DEPARTMENT OF INFORMATION TECHNOLOGY**


**VSIT** | Vidyalankar School of  
Information Technology  
NAAC ACCREDITED COLLEGE

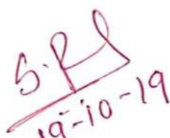
**CERTIFICATE**

This is to certify that the project entitled, "**Low Cost Health Monitoring System**",  
is bonafied work of **Ritik Rajesh Samanta** bearing Seat No: \_\_\_\_\_ submitted in  
partial fulfilment of the requirements for the award of degree of BACHELOR OF  
SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

  
Internal Guide

  
Coordinator

  
Internal Examiner

  
External Examiner

Date: 19/10/19



  
Principal

**TANTRA VIHAR**  
**2019-20**

**CERTIFICATE  
OF PARTICIPATION**

*This is to certify that*

***Ritik Samanta***

*of Vidyalankar School of Information Technology*

*has successfully exhibited the project titled*

***Low Cost Health Monitoring System***

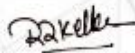
*in **Tantravihar 2019-20** Annual Project Exhibition  
held on 6<sup>th</sup> March 2020 and 13<sup>th</sup> March 2020.*



Ms. Pushpa Mahapatro  
Convener



Mr. Sagar Gaikwad  
Convener



Dr. Rohini Kelkar  
Principal



Mahatma Education Society's

PILLAI HOC COLLEGE OF ARTS, SCIENCE AND COMMERCE

PILLAI HOC EDUCATIONAL CAMPUS, RASAYANI, KHALAPUR, RAIGAD (DIST-410207)

(Accredited by NAAC)



Department of Information Technology & Computer Science

In Association with



Internal Quality Assurance Cell (IQAC) & IJSER

## Certificate

This is to certify that

Mr./Ms./Dr.

Ritik Samanta

of

Vidyalankar School of Information Technology

has participated/

presented a research paper titled Low Cost Health Monitoring System

in One Day National Conference on

'Emerging Trends in Computer Science & Information Technology' (NCEITSIT-2020) held on February 22, 2020.

Mr. Deepesh Jagdale  
Convener

Ms. Babitha Kurup  
Co-Convener

Dr. Lata Menon  
Principal



# ABSTRACT

A busy lifestyle leaves people with very little for regular health check-ups or visits to the doctor. At the same time, people wish that they would be able to monitor their own as well as the health of children, dependents, old parents alone at home, with the help of some basic health parameters. Certain conditions such as Atrial fibrillation that involves an irregular heartbeat, many a times go unnoticed leading later on to blood clots, stroke, heart failure and other heart-related complications. The RELY-AF registry reported that amongst the AF patients, valvular heart disease was most common in India (46.7%) and Africa (32.6%) and much less common in Eastern Europe (10.7%) and Western Europe (8.8%).

An increase in body temperature (fever) which tends to get ignored at times or in case of children alone at home, goes unnoticed could actually require timely intervention as it could be indicative of a minor or major infection that needs to be treated. Our project LOW COST HEALTH MONITORING SYSTEM proposes an Embedded solution that includes a light weight, portable wearable device to check heart rate and body temperature. The wearable device would continuously collect this data and send it with the help of a mobile app to the parents / caretakers. It would help parents to immediately contact the doctor or even admit to the hospital, their children / old parents. The mobile app additionally can then be used to track the person's heart rate and body temperature over a period of time.

**Keywords :** Heart rate, body temperature, health monitoring, low cost.

## ACKNOWLEDGEMENT

It gives us immense pleasure to express our sincere gratitude to those who are associated with our project LOW COST HEALTH MONITORING SYSTEM which is embedded/IOT prototype as a Part of the Course of BSc(IT) affiliated by the University of Mumbai.

We are grateful to our Principal **Dr.Rohini Kelkar** for her constant support and help as well as our Vice Principal **Mr.Asif Rampurawala** and Project Committee Head **Mrs.Pushpa Mahapatrao** to give us an opportunity to take up a Project.

We would like to thank our Project Guide **Mrs. Prachi Mahajan** who has been a constant source of motivation,encouragement and guidance. Her Valuable guidance and timely help has helped us in completing the project.

Above all our thanks to our family and friends for their help and motivation in the making of our project.

# DECLARATION

I hereby declare that the project entitled, “**Low Cost Health Monitoring System**” done at Vidyalankar School of Information Technology, has not been in any case duplicated to submit to any other universities for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

RITIK RAJESH SAMANTA

Name and Signature of the Student



## Table of Contents

Chapter 1 Introduction .....	13
1.1 Background .....	13
1.2 Objectives.....	14
1.3 Purpose, Scope , Applicability (Feasibility Study) .....	15
1.3.1 Purpose .....	15
1.3.2 Scope.....	15
1.3.3 Applicability.....	15
Chapter 2 Survey of Technologies .....	16
Chapter 3 Requirements and Analysis .....	18
3.1 Problem Definition .....	18
3.2 Requirement Specification.....	18
3.3 Planning and Scheduling .....	19
3.4 Software and Hardware Requirement .....	22
Chapter 4 System Design .....	24
4.1 Basic Modules .....	24
4.2 Data Design (Table Design) .....	24
4.3 Diagrams .....	25
4.3.1 Block Diagram .....	25
4.3.2 Use Case Diagram .....	26
4.3.3 Sequence Diagram .....	27
4.3.4 Activity Diagram .....	27
4.3.5 Component Diagram .....	28
4.3.6 Menu Tree.....	29
4.3.7 Event Table.....	29
4.3.8 User Interface Design.....	30
4.3.9 Test Cases Design .....	32
Chapter 5 Implementation and Testing .....	33
5.1 Implementation Approaches .....	33
5.2 Coding Details and Code Efficiency:.....	35
5.3 Testing approaches .....	70
5.3.1 Unit Testing .....	70
5.3.2 Integration Testing.....	71
5.4 Modification and Improvements .....	71
Chapter 6 Results and Discussion .....	72
6.1 Test Reports .....	72

6.2 User Documentation.....	73
Chapter 7 Conclusion .....	76
7.1 Conclusion.....	76
7.2 Limitation of the system .....	76
7.3 Future Scope of the Project .....	76
References .....	77
Bibliography .....	78
Website Used.....	78
Glossary.....	79
Summary .....	80
Further Reading .....	81
Plagiarism Report.....	82

# List of Tables

TABLE 2. 1 BETTER WAY TO PROGRAM NODEMCU 12.....	16
TABLE 2. 2 DEVELOPMENT BOARD .....	17
TABLE 2. 3 SERVER TO BE USED .....	17
TABLE 2. 4 APPLICATION.....	17
TABLE 3. 1 ALLOCATED TIME FOR EACH TECHNOLOGY .....	19
TABLE 4. 1 EVENT TABLE.....	29
TABLE 5. 1 IMPLEMENTATION APPROACHES .....	33
TABLE 6. 1 TEST REPORTS.....	72

# List of Figures

FIGURE 1 WATERFALL MODEL.....	20
FIGURE 2 GANTT CHART .....	21
FIGURE 3 PERT CHART.....	21
FIGURE 4 ESP8266 NODEMCU 12 DEVELOPMENT BOARD .....	22
FIGURE 5 DHT11 TEMPERATURE SENSOR.....	22
FIGURE 6 APDS9008 – PULSE SENSOR MODULE .....	22
FIGURE 7 BLOCK DIAGRAM.....	25
FIGURE 8 USE CASE DIAGRAM.....	26
FIGURE 9 SEQUENCE DIAGRAM .....	27
FIGURE 10 ACTIVITY DIAGRAM .....	27
FIGURE 11 COMPONENT DIAGRAM.....	28
FIGURE 12 MENU TREE .....	29
FIGURE 13 USER INTERFACE DESIGN .....	31

# Chapter 1 Introduction

The Project “Low Cost Health Monitoring System” is a project made by keeping an idea and vision of making a device that can help any common people to check their own or anyone’s health by checking their health parameters within a very low price. It is an embedded and software project that operates as a method to monitor people’s health readings using an application. It is essentially a system to monitor people’s health using a hardware device and mobile application.

It is a combination of hardware device with bunch of sensors connected to it to monitor readings from the person wearing it and using internet we send the data to a server and then fetch those readings in the application to display it and if we noticed any abnormality we can help the person on time by sending help even if we are not physically present over there

## 1.1 Background

A busy lifestyle leaves people with very little for regular health check-ups or visits to the doctor. At the same time, people wish that they would be able to monitor their own as well as the health of children, dependents, old parents alone at home, with the help of some basic health parameters. If any person is having a normal fever but it needs to be checked on regular intervals, then it is difficult to do so in our regular work days. That’s why for managing health issues we can make use of technologies for having a better way of having a health check by monitoring it. Heart Diseases are one of the main common diseases among many peoples.

In this system we plan to design a hardware device consisting of sensors with it that connects to the internet and starts collecting data readings from the person wearing it and sends to the server and that data is being displayed by using a mobile application .The application is only being used by the authorized user or any family member to monitor and have a time to time check over the persons’s health condition.

## 1.2 Objectives

The Project “Low Cost Health Monitoring System” is an embedded system which is combination of hardware device with bunch of sensors connected to it to monitor readings from the patient and using internet we send the data to a server and then fetch those readings in the application to display it. and if noticing an abnormality the application generates an emergency response and sends an notification to the caretaker.

- It will help to monitor patient’s health by capturing reading
- Care taker will be get notified about any abnormality in patient’s health the application will generates an emergency response and sends an alert message to the caretaker.
- This will help to know about the patient health before any crisis happens.
- The application will give access to data only to authorized person can be used by any of the family member or any care taker if kept for the patient.

## **1.3 Purpose, Scope , Applicability (Feasibility Study)**

### **1.3.1 Purpose**

The main idea behind making this project was to monitor the patient's health by capturing readings and proper view on his current health. Because being there for the patient everytime is not possible in daily scheduled life .That is why using this application makes it easier to check patient's health .Does the patient requires any care at the moment so we can get help for them and with this application we can view the readings from anywhere using the Internet and can get alert messages.

### **1.3.2 Scope**

The hardware to be created for the purpose requires to be small and wearable. It can be worn by any person, at home or hospital. In hospital's or in private clinics using this device can ease the process of check-ups and reduce the queues. Accessing the data requires a mobile application. With such systems you can check the pulse rate by sitting at home only and dont need to go anywhere else and you can check it anytime.

### **1.3.3 Applicability**

The mobile application is an important part as the device will collect data for any individual's health by capturing readings and for viewing that collected data from the server this mobile application will be used with the hardware device together. The data will help us to know about patient's health whether future examinations need to be done or not.

### **FEASIBILITY STUDY**

Feasibility study decides whether the proposed system is feasible by considering the economical and operational factors.

#### **ECONOMIC FEASIBILITY:**

The project is economically feasible. It does not require any financial investment and it is possible to complete the project on time. The cost include :

- Cost for full system investigation.
- Cost of hardware and software.
- Maintenance cost.
- Cost of development tools.

The cost of development is nothing compared to the financial benefits of the application.

#### **OPERATIONAL FEASIBILITY :**

Different operational like manpower, time are considered . Our project is completely feasible considering the given time.



## Chapter 2 Survey of Technologies

Which is the better way to program for Nodemcu 12 e?



**Table 2. 1 Better Way To Program Nodemcu 12 e**

Arduino IDE	MicroPython
1) The Arduino Integrated Development Environment(IDE) is a cross-platform application for Windows , macOS,Linux that is written in the programming language Java.	1)Python is comparatively easier to code for embedded and iot products.
2) It is used to write and upload programs to Arduino compatible boards,but also with the help of 3 <sup>rd</sup> party cores, other vendor development boards.	2) The MicroPython software supports the ESP8266 chip itself and any board should work.
3) The Arduino IDE supports the languages C and C++ using special rules of code structuring.The Arduino IDE supplies a software library from the wiring project ,which provides many common input and output procedures.	When it comes to MicroPython, it only comes with a small subset of the standard Python library, but it does come with a set of modules to control GPIOs, make networks connections and much more

**Table 2. 2 Development board**

<b>NodeMcu 12</b>	<b>Raspberry Pi</b>
1) NodeMCU is an open source IOT Platform	1) The <b>Raspberry Pi</b> is a low cost, creditcard sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse.
2) It includes firmware which runs on the ESP8266 Wi-Fi Soc from ESpressif Systems, and hardware which is based on the ESP-12 module.	2) It supports with multiple ports and 40 Gpio pins.

**Table 2. 3 Server to be used**

<b>Thingspeak</b>	<b>Adafruit IO</b>
1) ThingSpeak is an open source Internet of Things(IoT) applications and API to store and retrieve data from things using the HTTP and MQTT protocol over the Internet or via a Local Area Network	1) Adafruit IO is a system that makes data useful. Our focus is on ease of use, and allowing simple data connections with little programming required
2) ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates.	2) IO includes client libraries that wrap our REST and MQTT APIs. IO is built on Ruby on Rails, and Node.js.
3) ThingSpeak server is an open data platform and API for the Internet of Things that enables you to collect, store, analyze, visualize and act on data from sensors.	3) The Adafruit IO client libraries greatly simplify interacting with the server
4) Thingspeak provides its own inbuilt applications to check the data	4) It shows data on adafruit io website.

**Table 2. 4 Application**

<b>Android studio</b>	<b>Mit app inventor</b>
1) The Adafruit IO client libraries greatly simplify interacting with the server.	1) MIT app Inventor is basically a simple Framework which can be used to develop small to medium apps.
2) It have all the functionalities and tools for creating any kind of App.	2) It does not support all the functionalities that are present in the Android Studio.
3) This is the tool which developers and professionals use to create professional looking Apps.	3) Another main thing that app Inventor lacks is debugging functionalities.

## Chapter 3 Requirements and Analysis

### 3.1 Problem Definition

The problem with people is about managing time for the focusing on own or any individual's health or to take them to hospitals on time, or being there with them.

Problem that occurs with this is they can not have proper way to take care of the family member or any person.

Time issue : Giving time every day for health checkups is very difficult.

Doctor's Availability : Many time doctors and medical staffs are not available in every locations.

### 3.2 Requirement Specification

#### ❖ Functional Requirements

The Patient Care system will have the following functional requirements:

- It will help to keep an proper eye on patient's health using the multiple parameters including pulse rate and body temperature.
- Its a light weight wearable which is portable and easy to take it anywhere.
- The application will collect readings of particular patient with particular Id.
- The application will help to access data from anywhere using Internet as a medium.

#### ❖ Non-Functional Requirements

- **Performance** – Response for the readings will be fast and quick update for it.
- **Availability** – The system needs to be online for making it service available for everytime.
- **Reliability** – Reading provided by the sensors should be accurate with less deflections.
- **Usability & Regularity** – The application should be able to handle the data properly.

### 3.3 Planning and Scheduling

In order to successfully complete our project, it was necessary for us to learn various technologies.

For few technologies we had to start from scratch.

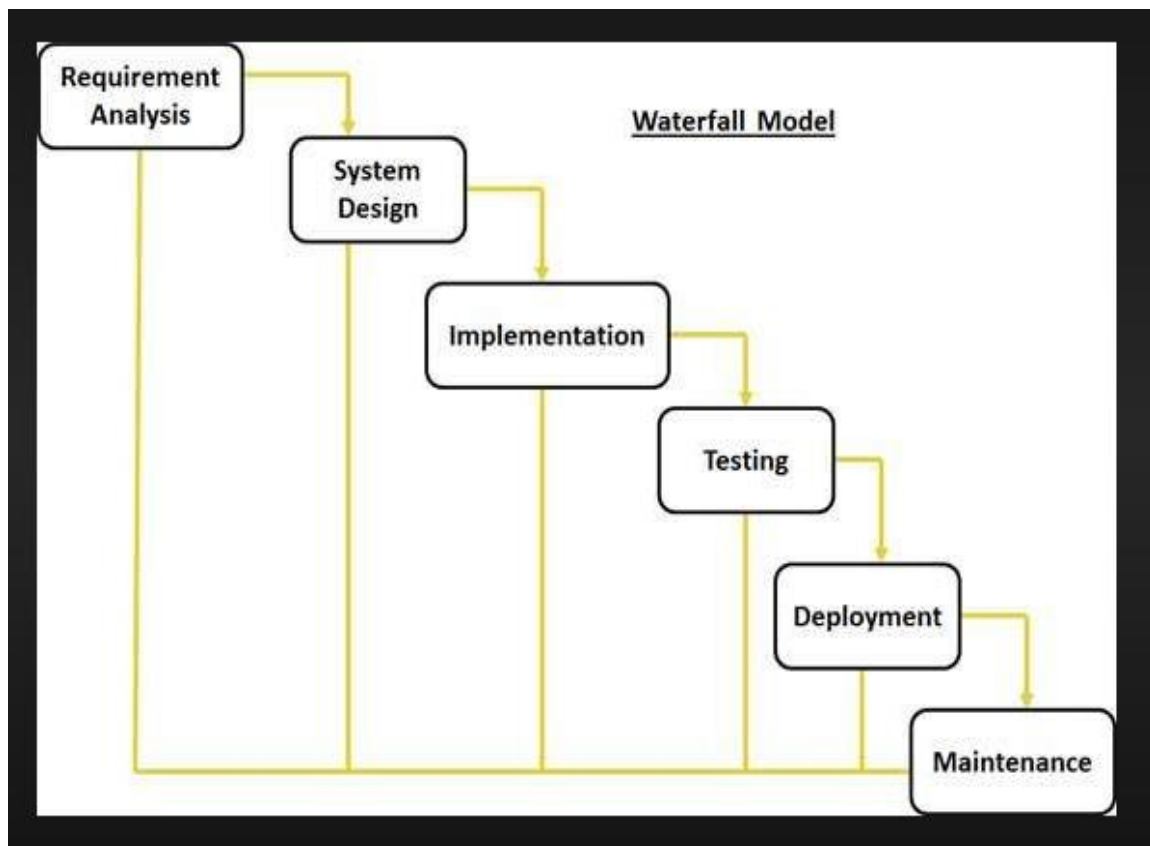
We allocated time for each technology according to their difficulty level.

	Technology	Average time allocated for learning
<b>1</b>	NodeMcu 12 Board <ul style="list-style-type: none"><li>• Study about pin diagram</li><li>• Working of the board</li><li>• Connectivity with internet and server</li></ul>	2-3 weeks
<b>2</b>	Arduino IDE <ul style="list-style-type: none"><li>• Knowing about the IDE</li><li>• Running the files in NodeMcu</li><li>• Complete coding for sensors</li><li>• Importing libraries</li></ul>	3-4 weeks
<b>3</b>	Server <ul style="list-style-type: none"><li>• Search for proper platform for supporting nodemcu and arduino ide code</li><li>• Use of thingspeak services for storing data</li></ul>	3-4 weeks
<b>4</b>	Application <ul style="list-style-type: none"><li>• Understanding the requirement for application</li><li>• Application design</li><li>• Records fetching</li></ul>	2-3 weeks

**Table 3. 1 Allocated Time For Each Technology**

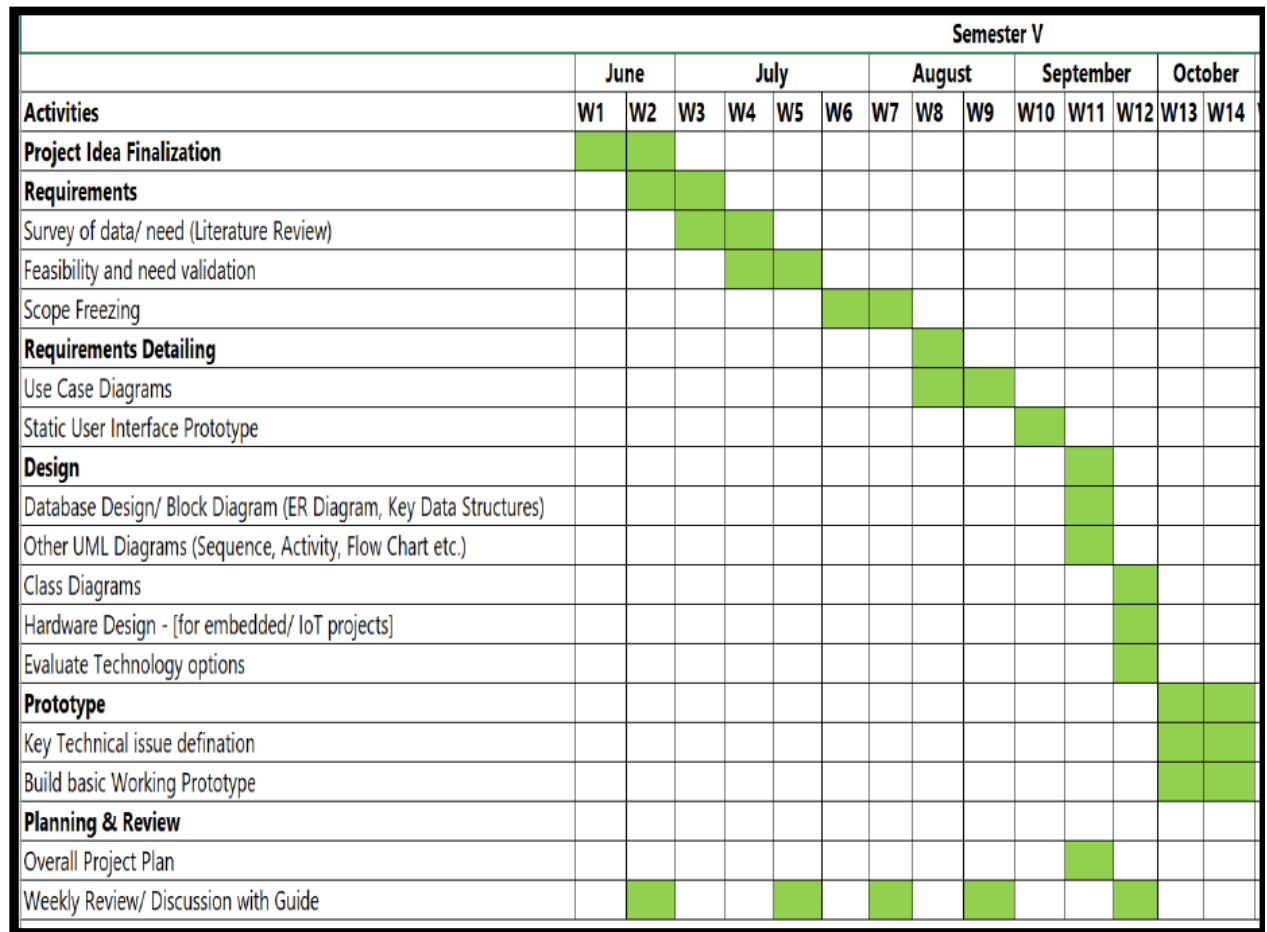
We will be using Waterfall model for implementing our project.  
Significance of using waterfall model:

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood.



**Figure 1 WaterFall Model**

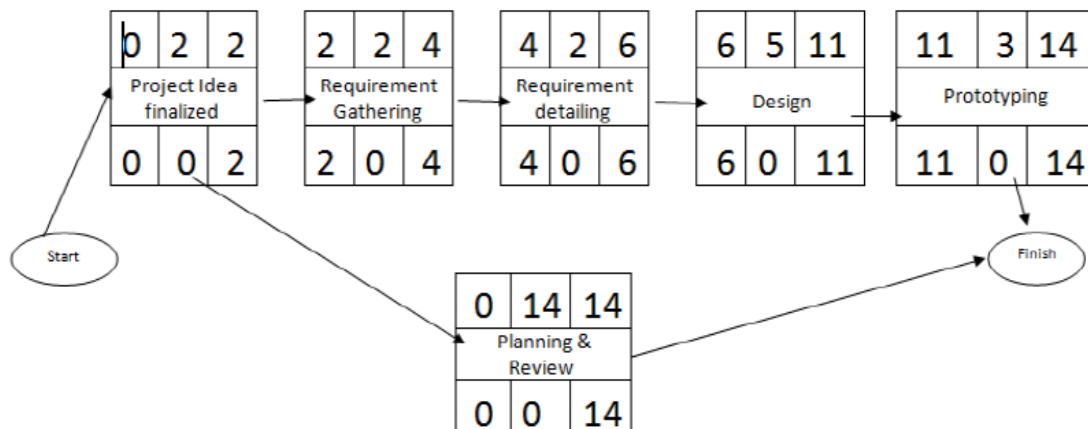
## Gantt Chart:



**Figure 2 Gantt Chart**

## Pert Chart:

A PERT chart is a project management tool that provides a graphical representation of a project's timeline. The Program Evaluation Review Technique (PERT) breaks down the individual tasks of a project for analysis.



**Figure 3 Pert Chart**

## 3.4 Software and Hardware Requirement

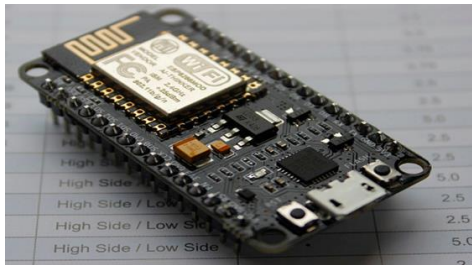
Requirements for developer

### Software Requirements:

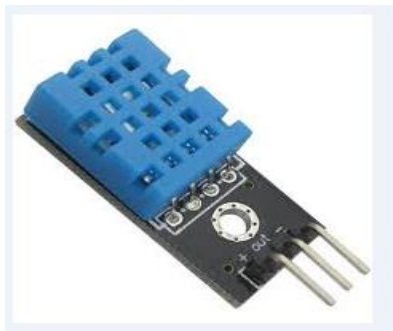
- 1)Cordova
- 2)Web Server like thingspeak
- 3)Brackets editor
- 4)Arduino IDE 1.8
- 5)Apache PhoneGap build

### Hardware Requirements:

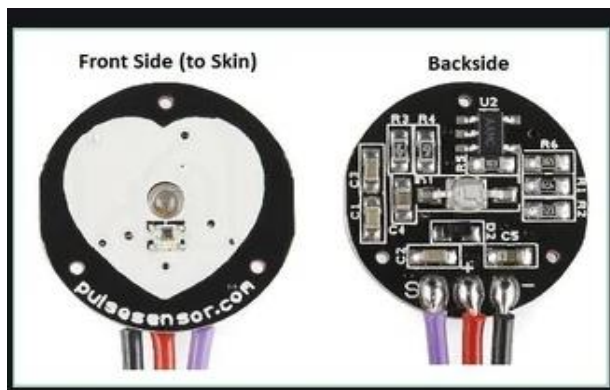
**Figure 4 ESP8266 NodeMcu 12 Development Board**



**Figure 5 Dht11 Temperature Sensor**



**Figure 6 APDS9908 – Pulse Sensor Module**





## **Software and Hardware requirements for a developer:**

For software there are many ways to go for other languages for coding the hardware device but better way is to go for Arduino IDE as the sensors do match with it and we can get inbuilt libraries in the IDE and even we can add on multiple extra libraries directly importing it.

For storing the readings either we can use Adafruit.io ,Amazon Web Services , FireBase,Xampp Server for LAN and ThingSpeak.

For making the applications best way is either to use Android studio ,MIT App Inventor,Cordova.

## Chapter 4 System Design

### 4.1 Basic Modules

**1.Wearable :** The wearable is one module of hardware in our project . It consist of the use of NodeMcu 12 development board connected with temperature sensor and pulse rate sensor with using internet to send readings.

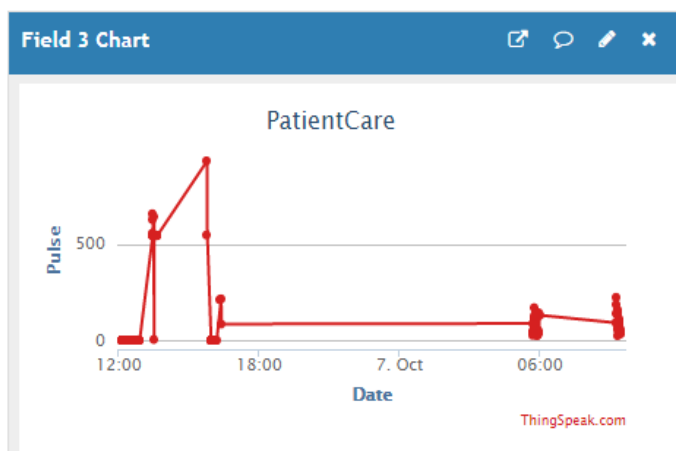
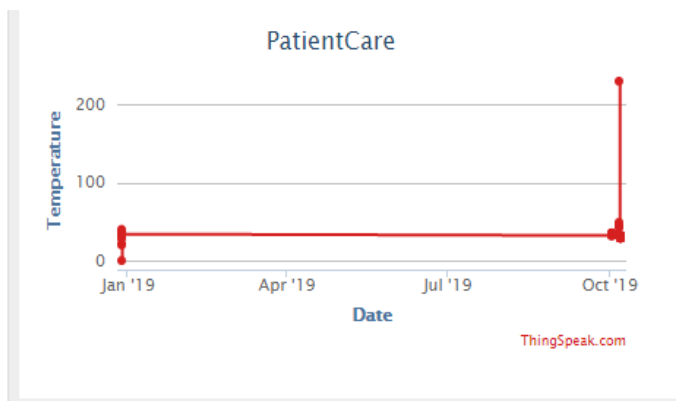
**2.Application :** This is the application part through which user can check the patient's health readings which will be fetched from the wearable device.

### 4.2 Data Design (Table Design)

The server where we are going to store the data is by using Thingspeak.

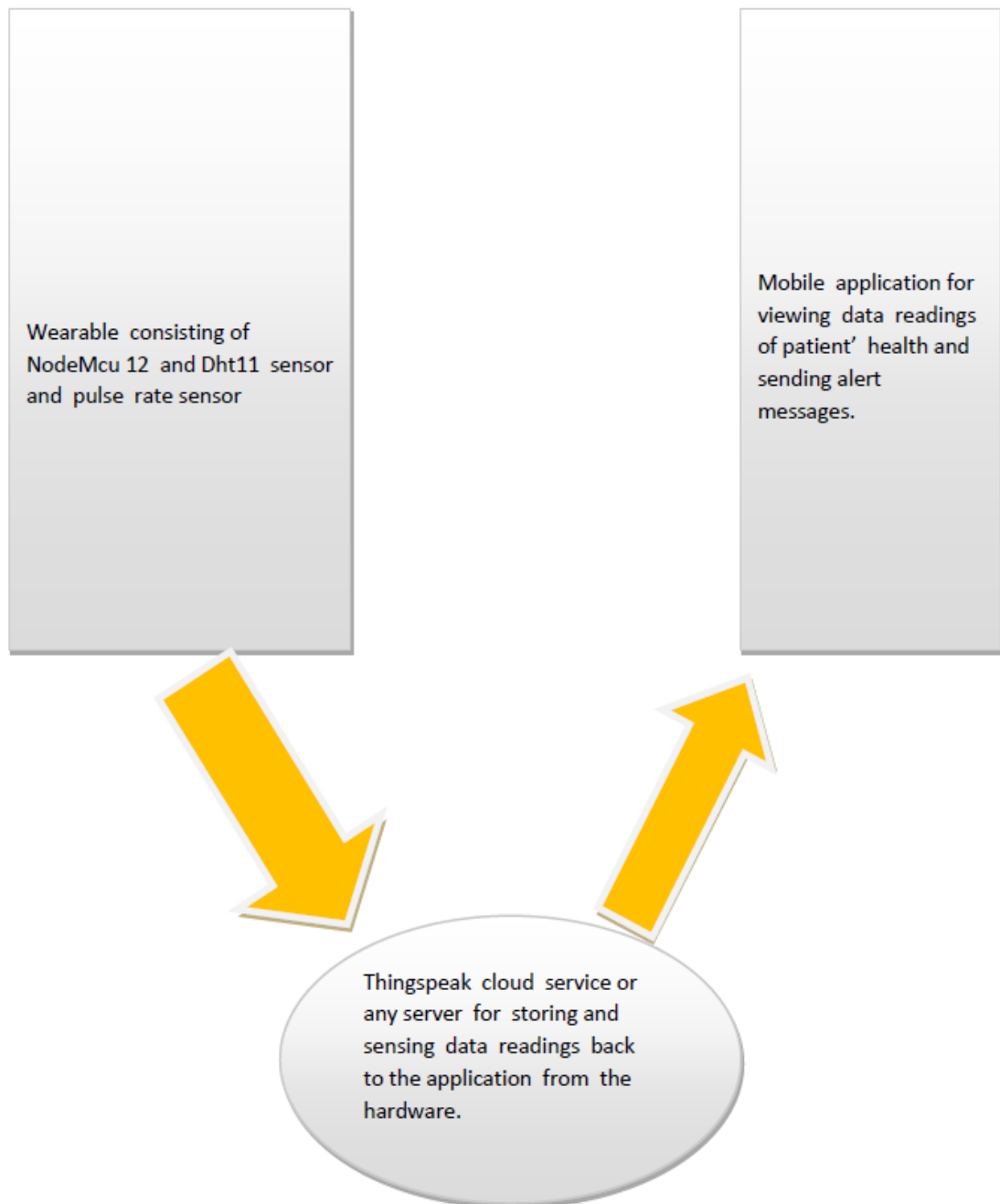
In our project we are using Analog Pulse Rate sensor and Digital Temperature sensor But the readings are going to be in numerical format only.

The data readings right now we are capturing are Pulse rate, Body Temperature using sensors respectively.



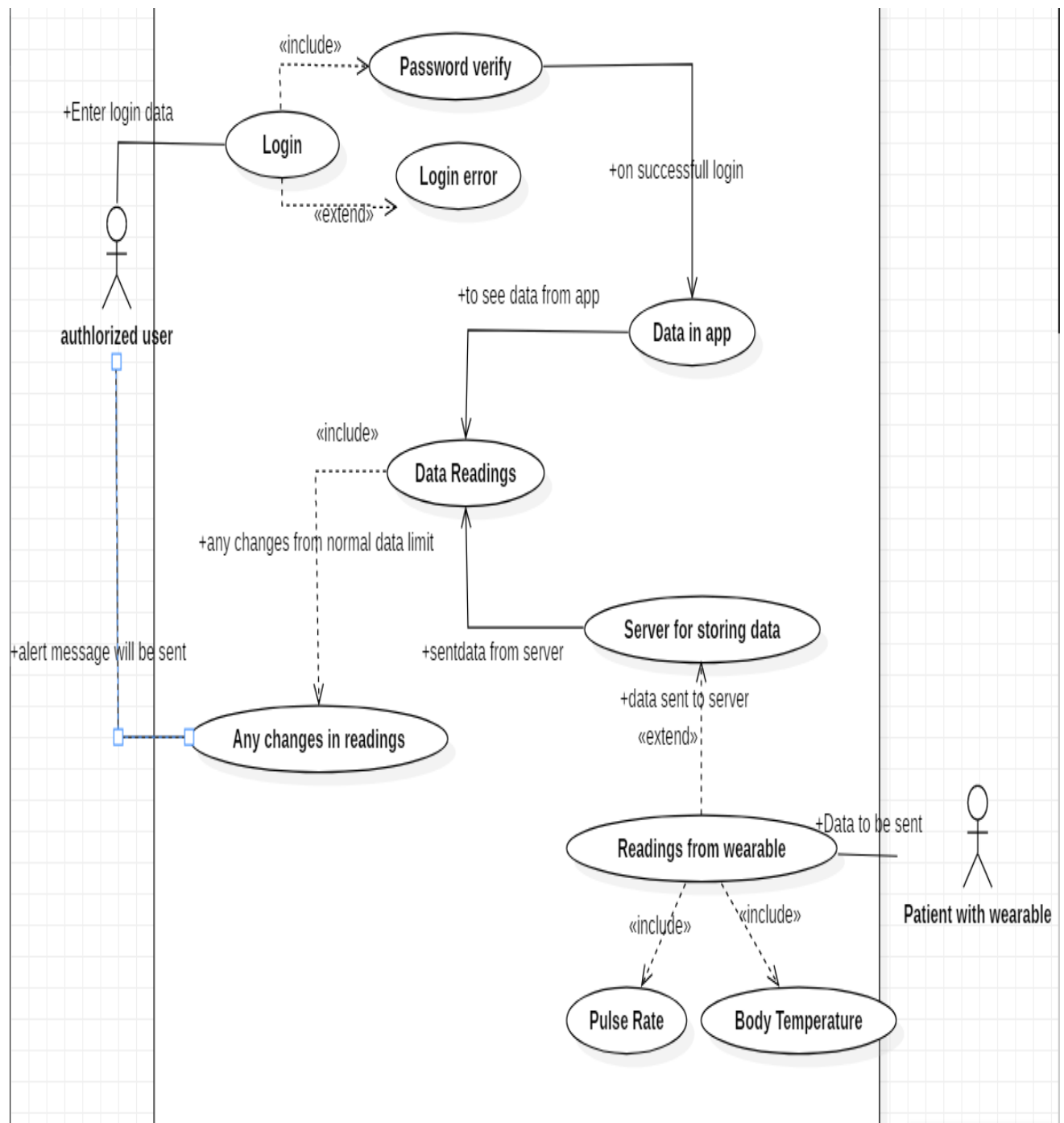
## 4.3 Diagrams

### 4.3.1 Block Diagram



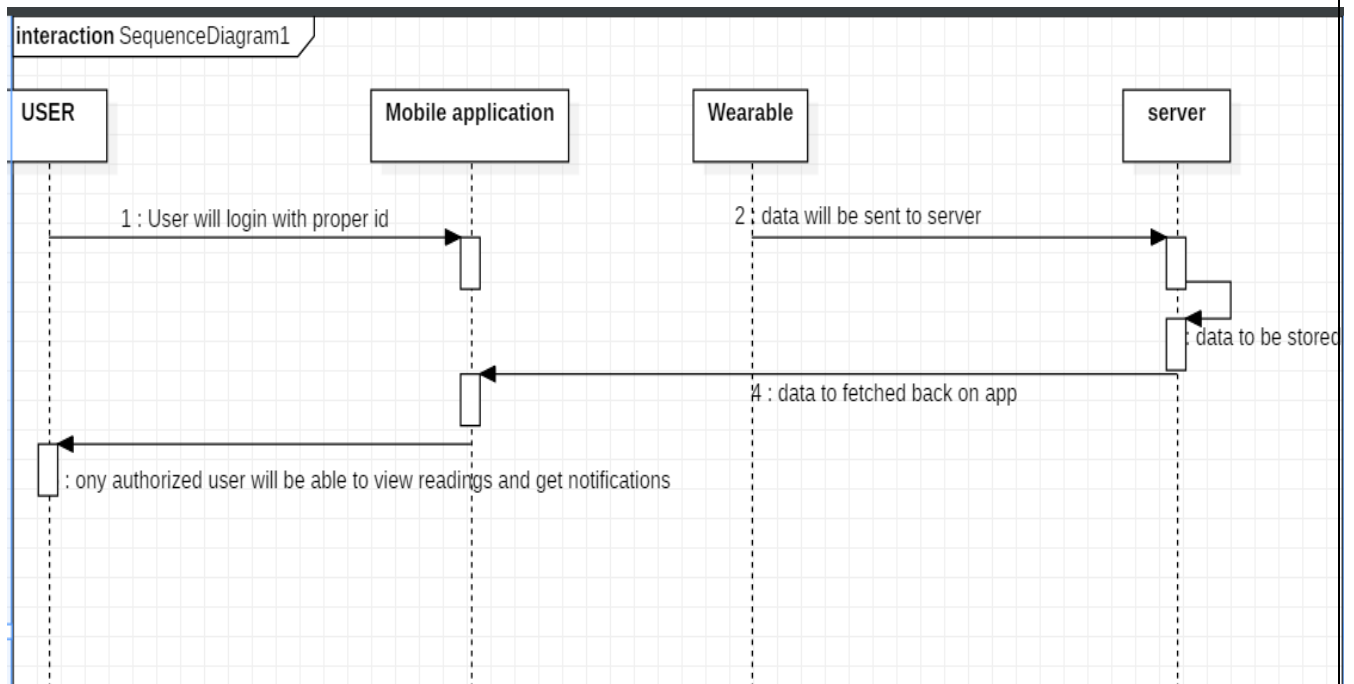
**Figure 7 Block Diagram**

### 4.3.2 Use Case Diagram



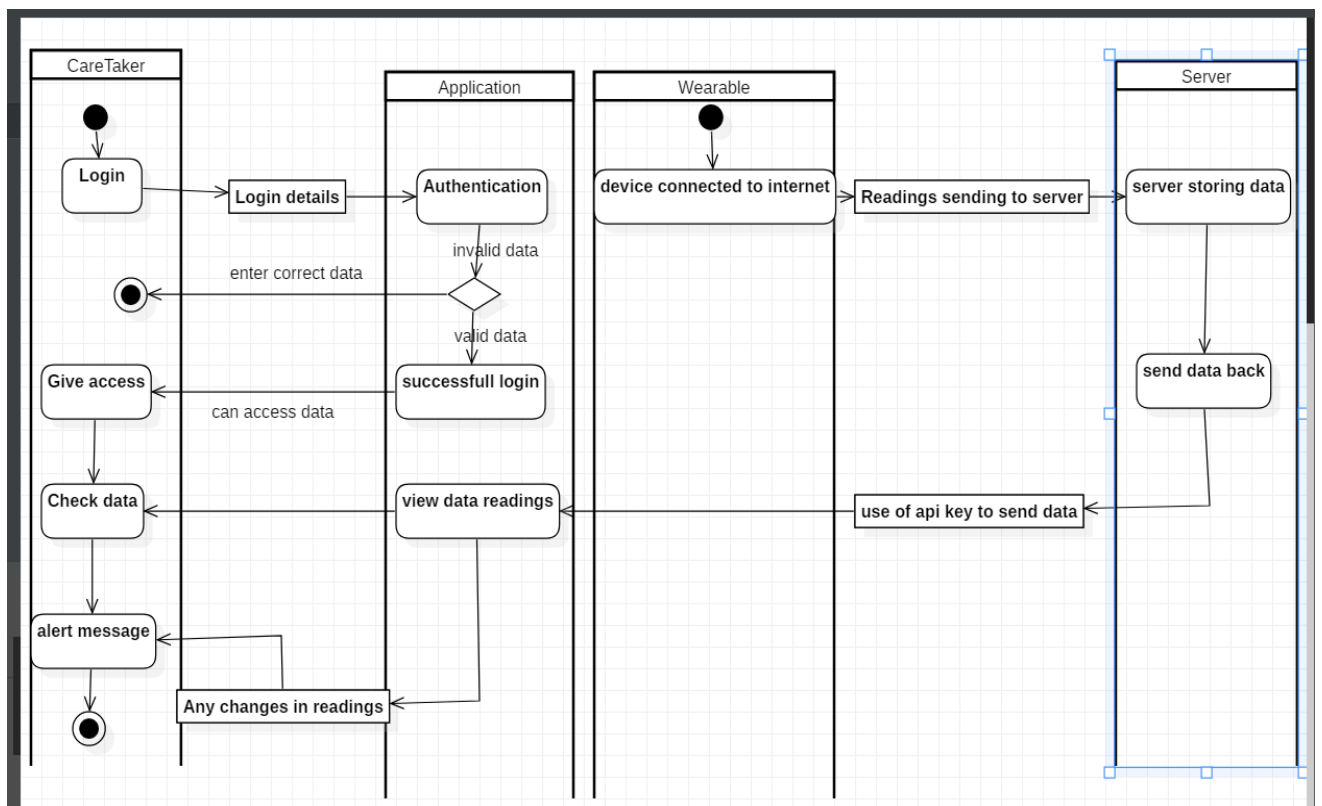
**Figure 8 Use Case Diagram**

### 4.3.3 Sequence Diagram



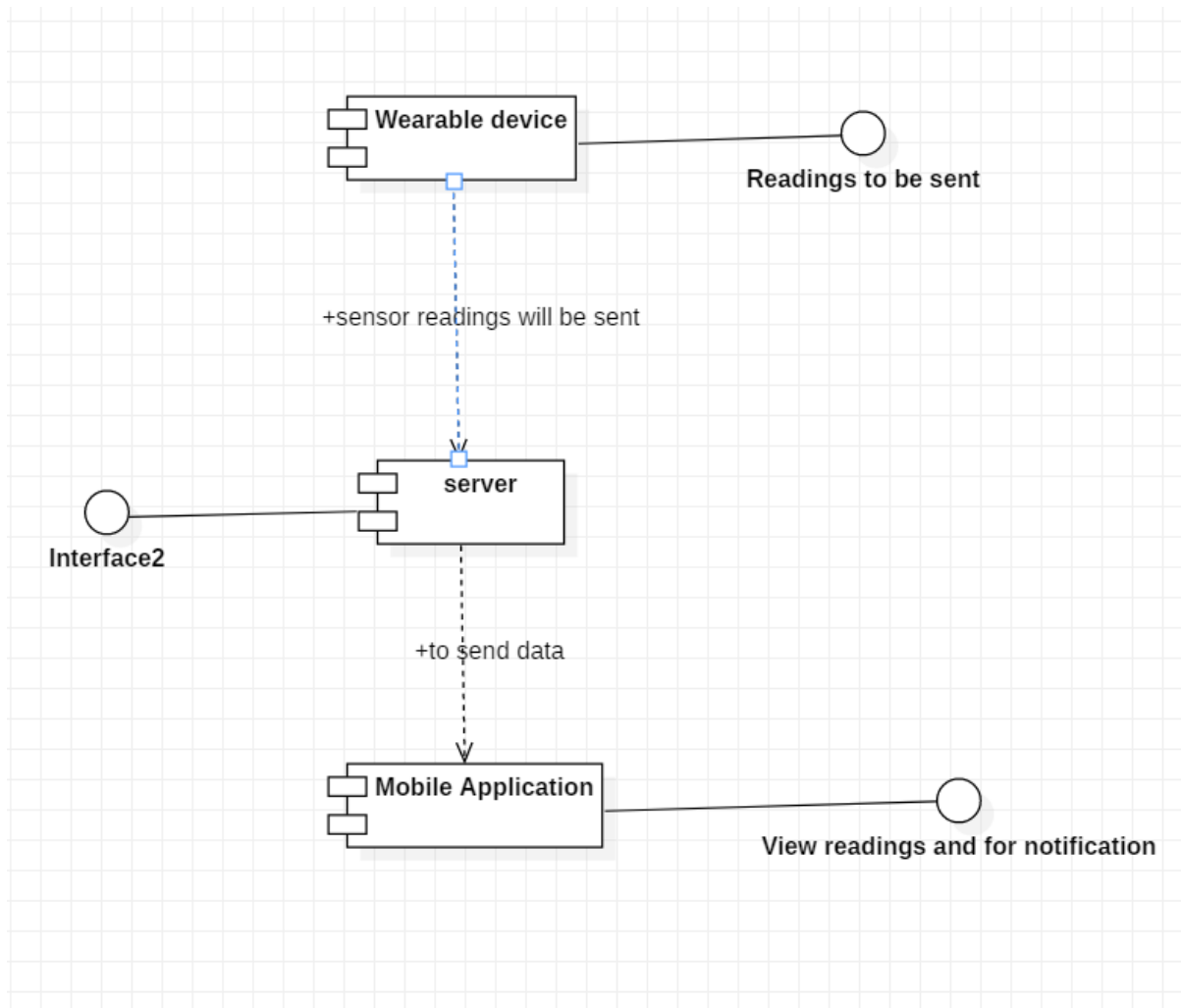
**Figure 9 Sequence Diagram**

### 4.3.4 Activity Diagram



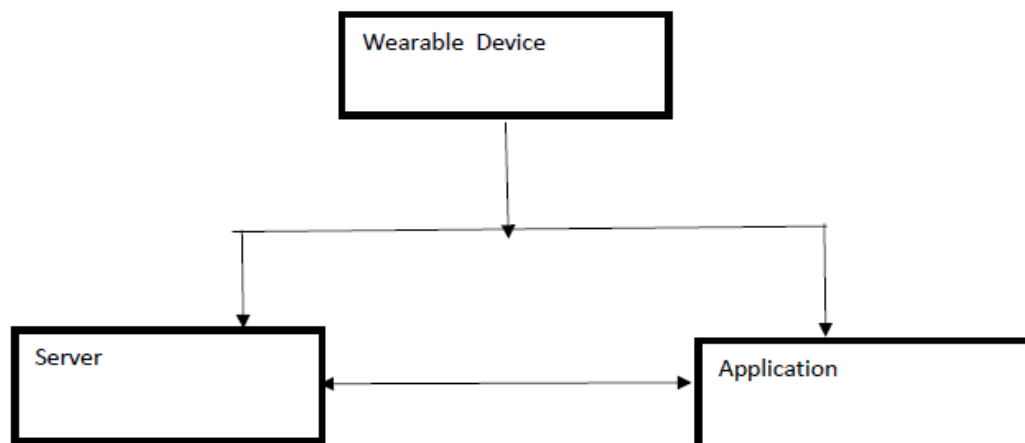
**Figure 10 Activity Diagram**

### 4.3.5 Component Diagram



**Figure 11 Component Diagram**

#### 4.3.6 Menu Tree



**Figure 12 Menu Tree**

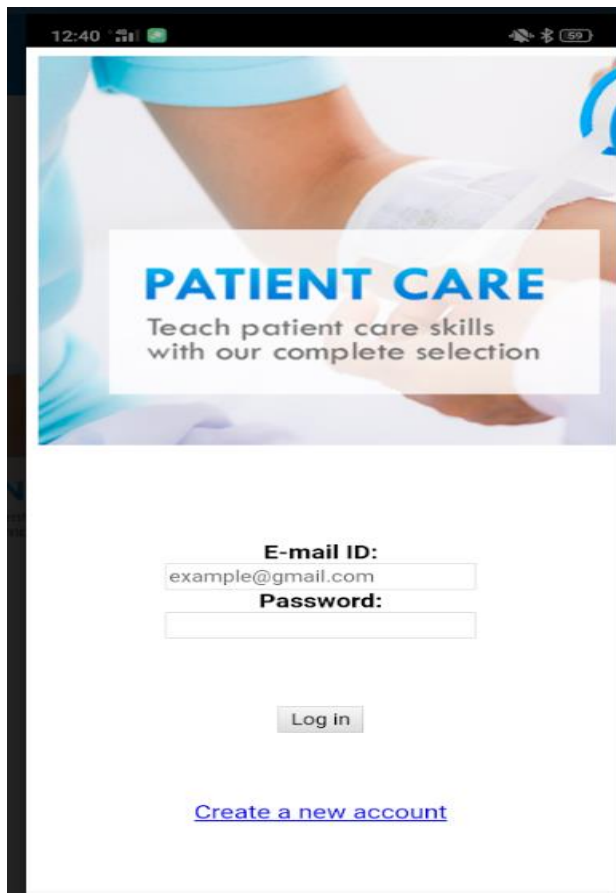
#### 4.3.7 Event Table

No	Event	Trigger	Activity	Source	Response	Destination
1	Login	Enter login credentials	Successful login	User	See previous data	-----
2	Data Generate	Give internet to wearable	Device Connected to internet	User	After connecting to internet data will generate	server
3	Record	Storing in server	Data stored in server successfully	Server used	Now can see readings	server
4	View	Use of api key to fetch data	Data to display in app	Server	Viewing data in the app	Mobile app

**Table 4. 1 Event Table**



### 4.3.8 User Interface Design



12:40 59

**PATIENT CARE**  
Teach patient care skills  
with our complete selection

**E-mail ID:**

**Password:**

[Create a new account](#)



12:40 59

**PATIENT FORM**

**Name:**

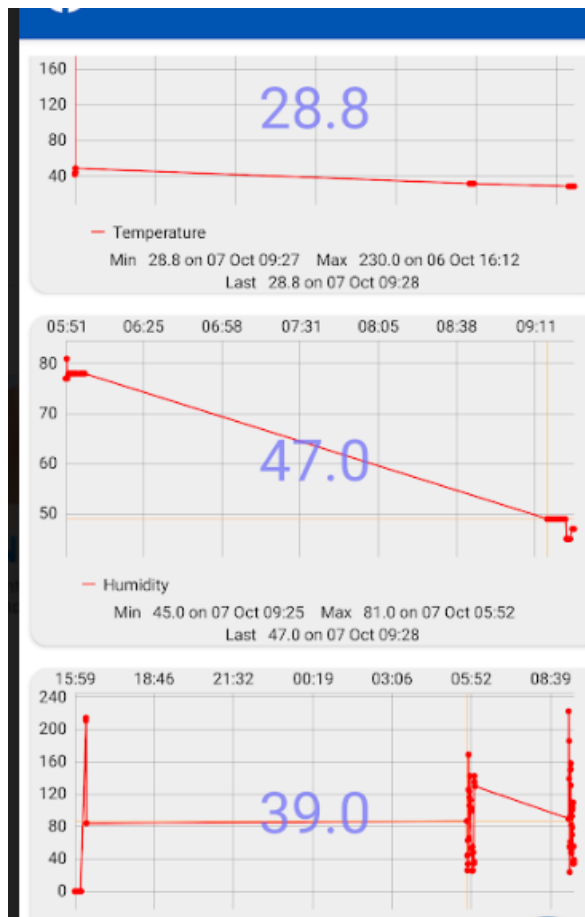
**E-mail ID:**

**CareTakerID:**

**Mobile Number 1:**

**Mobile Number 2:**

**Gender:** ☒ Male ☐ Female ☐ Other



**Figure 13 User Interface Design**

#### 4.3.9 Test Cases Design

Test Case	Input	Expected Output
Data Generate	Wearable device	Output as readings
Record	Data sent to server	Record storing at server
View	Server	Readings in mobile app

## Chapter 5 Implementation and Testing

### 5.1 Implementation Approaches

Semester VI													
Implementation													
Table Creation (if back-end database)													
Module 1													
Development													
Unit Testing													
Module 2													
Development													
Unit Testing													
System Testing													
Test case creation													
Integration Testing													
Acceptance Testing													
Planning & Review													
Overall Project Plan													
Weekly Review/ Discussion with Guide													

**Table 5. 1 Implementation Approaches**

Sr. No.	Implementation Plan	Action					
1.	Module	1.Wearable Device 2.Mobile Application					
2.	Percentage Completed	<table><tr><td>1.Wearable Device</td><td>100%</td></tr><tr><td>2.Mobile Application</td><td>100%</td></tr></table>		1.Wearable Device	100%	2.Mobile Application	100%
1.Wearable Device	100%						
2.Mobile Application	100%						
3.	Status	<table><tr><td>1.Wearable Device</td><td>Completed</td></tr><tr><td>2.Mobile Application</td><td>Completed</td></tr></table>		1.Wearable Device	Completed	2.Mobile Application	Completed
1.Wearable Device	Completed						
2.Mobile Application	Completed						
4.	Day Started	10 <sup>th</sup> December 2019					
5.	Day to be Completed	15 <sup>th</sup> February 2020					
6.	Actual Completion Date	18 <sup>th</sup> February 2020					
7.	Module Assignment	<table><tr><td>1.Wearable Device</td><td>Avdhoot Ambal</td></tr><tr><td>2.Mobile Application</td><td>Ritik Samanta</td></tr></table>		1.Wearable Device	Avdhoot Ambal	2.Mobile Application	Ritik Samanta
1.Wearable Device	Avdhoot Ambal						
2.Mobile Application	Ritik Samanta						
8.	Importance of Module	<table><tr><td>1.Wearable Device</td><td>Medium</td></tr><tr><td>2.Mobile Application</td><td>High</td></tr></table>		1.Wearable Device	Medium	2.Mobile Application	High
1.Wearable Device	Medium						
2.Mobile Application	High						

## 5.2 Coding Details and Code Efficiency:

### HARDWARE CODE DONE ON ARDUINO IDE

```
#define pulsePin A0

#include <ESP8266WiFi.h>

#include <DHT.h>

#define DHTPIN 14      //DHT11 is connected to GPIO Pin 2

String apiKey = "82Z4DCI923F6Z595"; // Enter your Write API key from ThingSpeak

const char* ssid = "Hackers"; // Enter your WiFi Network's SSID

const char* pass = "devil9721"; // Enter your WiFi Network's Password

const char* server = "api.thingspeak.com";

float humi;

float temp;


DHT dht(DHTPIN, DHT11);

WiFiClient client;


const int postingInterval = 10 * 1000;


int rate[10];

unsigned long sampleCounter = 0;

unsigned long lastBeatTime = 0;

unsigned long lastTime = 0, N;

int BPM = 0;

int IBI = 0;

int P = 512;

int T = 512;

int thresh = 512;

int amp = 100;

int Signal;

boolean Pulse = false;

boolean firstBeat = true;

boolean secondBeat = true;
```

```
boolean QS = false;
```

```
void setup() {
```

```
    Serial.begin(9600);
```

```
    delay(10);
```

```
    dht.begin();
```

```
    WiFi.begin(ssid, pass);
```

```
    while (WiFi.status() != WL_CONNECTED) {
```

```
        delay(500);
```

```
        humi = dht.readHumidity();
```

```
        temp = dht.readTemperature();
```

```
    }
```

```
}
```

```
void loop() {
```

```
    if (QS == true) {
```

```
        Serial.println("BPM: "+ String(BPM));
```

```
        QS = false;
```

```
    } else if (millis() >= (lastTime + 2)) {
```

```
        readPulse();
```

```
        lastTime = millis();
```

```
    }
```

```
}
```

```
void readPulse() {
```

```
    Signal = analogRead(pulsePin);
```

```
    sampleCounter += 2;
```

```
    int N = sampleCounter - lastBeatTime;
```



```
detectSetHighLow();
```

```
if (N > 250) {
```

```
    if ( (Signal > thresh) && (Pulse == false) && (N > (IBI / 5) * 3) )
```

```
        pulseDetected();
```

```
}
```

```
if (Signal < thresh && Pulse == true) {
```

```
    Pulse = false;
```

```
    amp = P - T;
```

```
    thresh = amp / 2 + T;
```

```
    P = thresh;
```

```
    T = thresh;
```

```
}
```

```
if (N > 2500) {
```

```
    thresh = 512;
```

```
    P = 512;
```

```
    T = 512;
```

```
    lastBeatTime = sampleCounter;
```

```
    firstBeat = true;
```

```
    secondBeat = true;
```

```
}
```

```
}
```

```
void detectSetHighLow() {
```

```
    if (Signal < thresh && N > (IBI / 5) * 3) {
```

```
        if (Signal < T) {
```

```
            T = Signal;
```

```
        }
```

```
    }
```

```
    if (Signal > thresh && Signal > P) {  
        P = Signal;  
    }  
  
}
```

```
void pulseDetected() {  
    Pulse = true;  
    IBI = sampleCounter - lastBeatTime;  
    lastBeatTime = sampleCounter;  
  
    if (firstBeat) {  
        firstBeat = false;  
        return;  
    }  
    if (secondBeat) {  
        secondBeat = false;  
        for (int i = 0; i <= 9; i++) {  
            rate[i] = IBI;  
        }  
    }  
}
```

```
word runningTotal = 0;
```

```
for (int i = 0; i <= 8; i++) {  
    rate[i] = rate[i + 1];  
    runningTotal += rate[i];  
}
```

```
rate[9] = IBI;  
runningTotal += rate[9];  
runningTotal /= 10;  
BPM = 60000 / runningTotal;  
QS = true;  
if (client.connect(server, 80)) {
```

```
String body = "field3=";

    body += BPM;

    String sendData =
apiKey+"&field1="+String(temp)+"&field2="+String(humi)+"&field3="+String(BPM)+"\r\n\r\n";

    //Serial.println(sendData);

    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(sendData.length());
    client.print("\n\n");
    client.print(sendData);

    Serial.print("Temperature: ");

    Serial.print(temp);

    Serial.print("deg C. Humidity: ");

    Serial.print(humi);

    client.println("");
    client.print(body);

}

client.stop();

delay(postingInterval);

}
```

## MOBILE APPLICATION CODE

### 1)Login.html

[illegible]

**<label>Password :</label>**

**<input type="password" name="password" id="password"/><br><br>**

**<input type="button" value="Login" id="submit" onclick="validate()"/>**

**</form>**

**</body>**

**</html>**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta name="viewport" content="width=device-width, initial-scale=1">**

**<meta name="msapplication-TileColor" content="#2d89ef">**

**<meta name="theme-color" content="#ffffff">**

**<link rel="stylesheet"  
href="https://cdn.jsdelivr.net/bootstrap/3.3.6/css/bootstrap.min.css">**

**<style type="text/css">**

**.feed-row { transition: background-color 2s ease; }**

**.rgb-cell { transition: color 0.5s ease, background-color 0.5s ease; }**

**.glyphicon { vertical-align: text-top; }**

**/\* restore webkit search cancel (overridden by Bootstrap) \*/**

**input[type="search"]::-webkit-search-cancel-button { -webkit-appearance:  
searchfield-cancel-button; }**

**</style>**

**</head>**

**</html>**

## 2)Menu.html

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta name="viewport" content="width=device-width, initial-scale=1">**

**<title>HEALTH MONITOR</title>**

**<link rel="manifest" href="site.webmanifest">**

**<link rel="mask-icon" href="safari-pinned-tab.svg" color="#5bbad5">**

**<meta name="msapplication-TileColor" content="#2d89ef">**

**<meta name="theme-color" content="#ffffff">**

**<link rel="stylesheet"  
href="https://cdn.jsdelivr.net/bootstrap/3.3.6/css/bootstrap.min.css">**

**<script src="https://cdn.jsdelivr.net/jquery/1.12.0/jquery.min.js"></script>**

**<script  
src="https://cdn.jsdelivr.net/bootstrap/3.3.6/js/bootstrap.min.js"></script>**

**</head>**

**<body>**

**<nav class="navbar navbar-default">**

**<div class="container-fluid">**

**<div class="navbar-header">**

**<button type="button" class="navbar-toggle" data-toggle="collapse"  
data-target="#iotNavBar">**

**<span class="icon-bar"></span>**

**<span class="icon-bar"></span>**

**<span class="icon-bar"></span>**

**</button>**

**</div>**

**<div class="collapse navbar-collapse" id="iotNavBar">**

**<ul class="nav navbar-nav">**

**<li><a href="1.html">LIVE DATA</a></li>**

**<li><a href="index2.html">DATA READINGS</a></li>**

**</ul>**

**</div>**

**</nav>**

**<main role="main" class="container">**

**<div class="jumbotron">**

**<h1>Welcome to HEALTH MONITOR</h1>**

**</div>**

**</main>**

**</body>**

**</html>**

**<!DOCTYPE html>**

**<html>**

**<title>W3.CSS</title>**

**<meta name="viewport" content="width=device-width, initial-scale=1">**

**<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">**

**<style>**

**.mySlides {display:none;}**

**</style>**

```
<body>

<div class="w3-container">

  <h2></h2>

</div>

<div class="w3-content w3-display-container">

<div class="w3-display-container mySlides">

  <div class="w3-display-topright w3-large w3-container w3-padding-16 w3-black">

    HEART ATTACK

  </div><br><br>

    <a href="heartdata.html"> </a>

  </div>

<div class="w3-display-container mySlides">

  <a href="feverdata.html"> </a>

  <div class="w3-display-topleft w3-large w3-container w3-padding-16 w3-black">

    FEVER

  </div>

</div>

<button class="w3-button w3-display-left w3-black" onclick="plusDvs(-
1)">&#10094;</button>

<button class="w3-button w3-display-right w3-black"
onclick="plusDvs(1)">&#10095;</button>

</div>

<script>

var slideIndex = 1;

showDvs(slideIndex);

function plusDvs(n) { showDvs(slideIndex += n);}
```



```
function showDivs(n) {  
    var i;  
    var x = document.getElementsByClassName("mySlides");  
    if (n > x.length) {slideIndex = 1}  
    if (n < 1) {slideIndex = x.length}  
    for (i = 0; i < x.length; i++) {  
        x[i].style.display = "none";  
    }  
    x[slideIndex-1].style.display = "block";  
}  
</script>  
</body>  
</html>
```

### 3)INDEX1.HTML

**<!DOCTYPE html>**

**<html height="100%">**

**<head>**

**<title></title>**

**<script type="text/javascript" src="js/jquery-1.12.1.js"></script>**

**<script type="text/javascript">**

```
function display_c() {  
    var refresh = 1000; // Refresh rate in milli seconds  
    mytime = setTimeout('display_ct()', refresh)  
}
```

```
function display_ct() {  
    var x = new Date()  
    document.getElementById('ct').style.fontSize = '30px';  
    document.getElementById('ct').style.color = '#0030c0';  
    document.getElementById('ct').innerHTML = x;  
    display_c();  
}
```

```
$(document).ready(function ()  
{  
    GetData();  
});
```

```
$(document).ready(function () {  
    GetDataa();  
});
```

```
function GetData()
```

```

{
    var url =
'https://api.thingspeak.com/channels/664409/fields/1.json?api_key=TIWWWE5MJP9J3QKU&results=2';

    $.ajax
    ({
        url: url,
        type: 'GET',
        contentType: "application/json",
        //dataType: "json",
        //crossDomain: true,
        success: function (data, textStatus, xhr) {
            $.each(data, function (i, item) {
                if (i == 'feeds') {
                    var ubound = item.length;
                    $('#txtField1').val(item[ubound - 1].field1);
                }
            });
        },
        error: function (xhr, textStatus, errorThrown) {
            alert(errorThrown);
        }
    });

    setTimeout(GetData, 10000);
}

```

```

function GetDataa() {
    var url =
'https://api.thingspeak.com/channels/664409/fields/3.json?api_key=TIWWWE5MJP9J3QKU&results=10';

    $.ajax
    ({
        url: url,
        type: 'GET',

```

```
contentType: "application/json",

//dataType: "json",

//crossDomain: true,

success: function (data, textStatus, xhr) {

    $.each(data, function (j, item) {

        if (j == 'feeds') {

            var ubound = item.length;

            $('#txtField3').val(item[ubound - 1].field3);

        }

    });

},

error: function (xhr, textStatus, errorThrown) {

    alert(errorThrown);

}

});
```

```
setTimeout(GetDataa, 10000);

}
```

</script>

<meta charset="utf-8" />

<style>

p { font-size: 30px; }

.units { font-size: 1.2rem; }

.dht-labels{

font-size: 20px;

vertical-align:middle;

padding-bottom: 15px;

}

</style>

</head>

<body onload=display\_ct();>

<center>

```

<span id='ct'></span><br ><br >

<p>
    TEMPERATURE (°C):

    <input id="txtField1" type="text"/><br><br>

    HEART RATE(BPM):

    <input id="txtField3" type="text" /><br><br>

    <a href="index2.html"><input type="button" name="VALUES" value="DATA
    READINGS"></a><br>

</p>

</center>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <meta name="msapplication-TileColor" content="#2d89ef">

    <meta name="theme-color" content="#ffffff">

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/bootstrap/3.3.6/css/bootstrap.min.css">

    <style type="text/css">

        .feed-row { transition: background-color 2s ease; }

        .rgb-cell { transition: color 0.5s ease, background-color 0.5s ease; }

        .glyphicon { vertical-align: text-top; }

        /* restore webkit search cancel (overridden by Bootstrap) */

        input[type="search"]::-webkit-search-cancel-button { -webkit-appearance: searchfield-cancel-
        button; }

    </style>

</head>

</html>

```

## **Index2.html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta name="viewport" content="width=device-width, initial-scale=1">**

**<meta name="msapplication-TileColor" content="#2d89ef">**

**<meta name="theme-color" content="#ffffff">**

**<link rel="stylesheet"  
href="https://cdn.jsdelivr.net/bootstrap/3.3.6/css/bootstrap.min.css">**

**<script src="https://cdn.jsdelivr.net/jquery/1.12.0/jquery.min.js"></script>**

**<script  
src="https://cdn.jsdelivr.net/bootstrap/3.3.6/js/bootstrap.min.js"></script>**

**<style type="text/css">**

**.feed-row { transition: background-color 2s ease; }**

**.rgb-cell { transition: color 0.5s ease, background-color 0.5s ease; }**

**.glyphicon { vertical-align: text-top; }**

**/\* restore webkit search cancel (overridden by Bootstrap) \*/**

**input[type="search"]::-webkit-search-cancel-button { -webkit-appearance:  
searchfield-cancel-button; }**

**</style>**

**<script type="text/javascript">**

**// setup time for updates**

```
var updates = null;

// setup IoT service details and defaults

var thingspeak = {};

// check local storage for thingspeak
if (localStorage.getItem('thingspeak')) {

    thingspeak = JSON.parse(localStorage.getItem('thingspeak'));

}
else {

    // set thingspeak
thingspeak['url'] = 'https://api.thingspeak.com';
thingspeak['channel'] = 664409;
thingspeak['read_api_key'] = 'TIWWWE5MJP9J3QKU';
thingspeak['results'] = 50;
thingspeak['start'] = false;

    // save to local storage
localStorage.setItem('thingspeak', JSON.stringify(thingspeak));

}

$(document).ready(function() {

    // set default inputs
$('#channel').val(thingspeak['channel']);
$('#read_api_key').val(thingspeak['read_api_key']);
```

```
$('#user_api_key').val(thingspeak['user_api_key']);
```

```
$('#tag').val(thingspeak['tag']);
```

```
if (thingspeak['start']) {
```

```
    $('#start').find(".glyphicon").removeClass("glyphicon-play").addClass("glyphicon-pause");
```

```
}
```

```
// when feed inputs change update params and clear output
```

```
$( ".feed-form" ).change(updateFeedParams);
```

```
// when channel inputs change update params
```

```
$( ".channel-form" ).change(updateChannelParams);
```

```
// start updates if play button is active
```

```
if (thingspeak['start']) {
```

```
    // start updates
```

```
    getUpdates();
```

```
    // check for new updates
```

```
    updates = setInterval('getUpdates()',5000);
```

```
}
```

```
// check if a user api key is available
```

```
if (thingspeak['user_api_key']) {
```

```
    // hide user key prompt and load channels
```



```
        $('#user-api-prompt').hide();
        refreshChannels();
    }
    else {
        // show channel config panel
        showConfigureChannelPanel();
    }

});

function updateFeedParams() {

    // update params
    thingspeak['channel'] = $('#channel').val();
    thingspeak['read_api_key'] = $('#read_api_key').val();

    // save to local storage
    localStorage.setItem('thingspeak', JSON.stringify(thingspeak));

    // clear output
    $('#header').html("");
    $('#output').html("");

}

function updateChannelParams() {

    if (!thingspeak['user_api_key'] && $('#user_api_key').val()) {
        // hide API key panel
        hideConfigureChannelPanel();
    }
}
```

```
}

// update params
thingspeak['user_api_key'] = $('#user_api_key').val();
thingspeak['tag'] = $('#tag').val();

// check if user api key is present
if (thingspeak['user_api_key']) {

    // hide prompt to enter key
    $('#user-api-prompt').hide();

    // refresh channels
    refreshChannels();

    // expand channel panel
    showChannelPanel();

} else {

    // clear display
    $('#channels-header').html("");
    $('#channels').html("");

    // show prompt to enter key
    $('#user-api-prompt').show();

    if (thingspeak['tag']) {
        // show API key panel
        showConfigureChannelPanel();
    }
}
```

```

    }

}

// save to local storage
localStorage.setItem('thingspeak', JSON.stringify(thingspeak));
}

function getUpdates() {

    // get the data with a webservice call

    $.getJSON( thingspeak['url'] + '/channels/' + thingspeak['channel'] +
'/feed.json?results=' + thingspeak['results'] + '&location=' + thingspeak['location']
+ '&status=' + thingspeak['status'] + '&api_key=' + thingspeak['read_api_key'],
function( data ) {

    // create table and headers if channel object exists
    if (data.channel) {

        // set headers
        $('#header').html('<b>' + data.channel.name + '</b>');

        // create table head and body if they do not exist
        if ($('#output').find('thead').length == 0) {

            // add head to output table
            $('#output').append("&<thead><tr></tr></thead>");

            // add date/time field as header
            $('#output').find('thead').append("<th>UPDATE TIME</th>");

```

```

        // add header for each field

        if (data.channel.hasOwnProperty('field1')) {
$('#output').find('thead').append("<th>TEMPERATURE</th>"); }

        if (data.channel.hasOwnProperty('field3')) {
$('#output').find('thead').append("<th>HEART RATE</th>"); }


        // add body to output table

        $('#output').append("<tbody></tbody>");

    }

}


// if the field1 has data update the page
if (data.feeds) {

    // go through each feed and add it to the top of the table if the row
does not exist

    $.each( data.feeds, function( i, feed ) {

        // add entry to table if it does not exist

        if ($('#output').find('#entry_' + feed.entry_id).length === 0) {

            // create a new blank row

            var $new_row = $("<tr id='entry_' + feed.entry_id + ' class='feed-
row'></tr>");

            // add time/date data to field

```

```
    $new_row.append("<td><time datetime='" + feed.created_at +  
    "'">" + feed.created_at + "</time></td>");
```

```
    // add field data to each field
```

```
    if (feed.hasOwnProperty('field1')) {  
$new_row.append(getTableElement(feed.field1)); }
```

```
    if (feed.hasOwnProperty('field3')) {  
$new_row.append(getTableElement(feed.field3)); }
```

```
    // add the row to the table
```

```
    $('#output').prepend($new_row);
```

```
    // set the background color to green
```

```
    $new_row.css('background-color', '#efe');
```

```
    // remove the background color after 500ms
```

```
    // css transitions are used with the feed-row class to make this
```

smooth

```
    setTimeout(function() {
```

```
        $new_row.css('background-color', "");
```

```
    }, 500);
```

```
    }
```

```
});
```

```
}
```

```
});
```

```
}
```

```
function getTableElement(content) {
```

```
// create a table element containing the given content
```

```
var $td = $('<td>' + content + '</td>');
```

```
// regex pattern to match RGB hex values
```

```
// remove preceding ^ to match colors within text
```

```
var rgbPattern = /^#([a-f\d]{2}){3}$/i;
```

```
// check content for a color
```

```
if (rgbPattern.test(content)) {
```

```
// extract that color
```

```
var color = rgbPattern.exec(content)[0];
```

```
// mark the cell as a rgb-cell for the CSS transitions
```

```
$td.addClass('rgb-cell');
```

```
// set the cell's background color to the hex color
```

```
$td.css('background-color', color);
```

```
// determine whether to set the font to white or black depending on the hex  
color
```

```
$td.css('color', useDarkFont(color) ? '#000' : '#fff');
```

```
// remove background and text color when hovering and restore when  
mouse exits
```

```
$td.hover(
```

```
function() {
```

```
        $td.css('background-color', "");

        $td.css('color', "");

    },

    function() {

        $td.css('background-color', color);

        $td.css('color', useDarkFont(color) ? '#000' : '#fff');

    });

}

return $td;

}
```

```
function refreshChannels() {

    var tagQuery = thingspeak['tag'] ? '&tag=' +
encodeURIComponent(thingspeak['tag']) : "";

    $.getJSON( thingspeak['url'] + '/channels.json?api_key=' +
thingspeak['user_api_key'] + tagQuery, function( data ) {

        // clear old channel data

        $('#channels').empty();

        if (data.length > 0) {

            // set channel header including tag if present

            $('#channels-header').html('Channels' + (thingspeak['tag'] ? ' with Tag: ' + thingspeak['tag'] : ''));

        } else {
```

```

        // set channel header including tag if present

        $('#channels-header').html('No Channels Found' + (thingspeak['tag'] ? '
with Tag: ' + thingspeak['tag'] : ''));

        // return so no table is drawn

        return;

    }

    // add a row for each channel

    $.each( data, function( i, channel ) {

        // create a new row

        var $new_item = $('<li class='list-group-item channel-row'></li>');

        $new_item.append($('<div class="text-muted" style="float:right">ID: '
+ channel.id + '<br />' + (channel.public_flag ? 'Public' : 'Private') + '</div>'))

        var $title = $('<span></span>');

        // create the channel name link element

        var $title_link = $('<h4 class="list-group-item-heading"><a href="#">' +
channel.name + '</a></h4>');

        // add an onClick event to the name link

        $title_link.click(function( event ) {

            // prevent navigation

            event.preventDefault();

            // find channel's read key

            var readKey = channel.api_keys.find(function ( key ) {

                return key.write_flag;

```



```
}}
```

```
// update forms with this channel's id and key
```

```
$('#channel').val(channel.id);
```

```
$('#read_api_key').val(readKey.api_key);
```

```
// update configuration
```

```
updateFeedParams();
```

```
// collapse channels panel
```

```
hideChannelPanel();
```

```
// start fetching updates
```

```
play();
```

```
}}
```

```
$title.append($title_link);
```

```
// add channel name to row
```

```
$new_item.append($title);
```

```
// wrap tag links and add icon
```

```
var $tag_item = $('<p class="list-group-item-text"><span  
class="glyphicon glyphicon-tags" />&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</p>');
```

```
$tag_item.append(listTags(channel.tags));
```

```
// add placeholder if no tags are present
```

```
if (channel.tags.length === 0) {
```

```
    $tag_item.append('<span class="text-muted">None</span>')
```

```

    }

    // add tag links to row

    $new_item.append($tag_item);

    if (channel.description && channel.description !== "") {

        $new_item.append($('<div class="text-muted">' +
channel.description + '</div>'));

    }

    // add the row to the table

    $('#channels').append($new_item);

});

});
}

function listTags(tags) {

    // create the list container

    var $list = $('<span></span>');

    // iterate through each tag

    $.each(tags, function ( i, tag ) {

        // create a link with the tag's name

        var $link = $('<a href="#">' + tag.name + '</a>');

        // make the tag bold if it's the currently searched tag

        if (tag.name.toUpperCase() === thingspeak['tag'].toUpperCase()) {

            $link.css("font-weight","Bold");

```

```

    }

    // add an onClick event that filters the user's channels by tag
    $link.click(function ( event ) {

        event.preventDefault();

        $('#tag').val(tag.name);
        updateChannelParams();

    });

    // add the tag to the list with a comma if it isn't the last tag
    $list.append($link).append($('<span>' + (i < tags.length - 1 ? ', ' : '') +
'</span>'));

    })

    return $list;
}

// tweaked from https://stackoverflow.com/questions/1855884/determine-font-
color-based-on-background-color

function useDarkFont(color) {

    // regex to separate RGB components
    var result = /^#?([a-fd]{2})([a-fd]{2})([a-fd]{2})$/i.exec(color);

    // Counting the perceptive luminance - human eye favors green color...
    var a = 1 - ( 0.299 * parseInt(result[1], 16) + 0.587 * parseInt(result[2], 16) +
0.114 * parseInt(result[3], 16)) / 255;

    return a < 0.5;
}

```

```
// start / stop updates

function playPause(button) {

    // switch button state and start or stop updates

    if ($(button).find(".glyphicon").hasClass("glyphicon-pause")) {

        pause();

    }

    else {

        play();

    }

}


function pause() {

    // show play button state

    if ($("#start").find(".glyphicon").hasClass("glyphicon-pause")) {

        $("#start").find(".glyphicon").removeClass("glyphicon-pause").addClass("glyphicon-play");

    }

    // update params

    thingspeak['start'] = false;

    // stop updates

    clearInterval(updates);

    // save to local storage

    localStorage.setItem('thingspeak', JSON.stringify(thingspeak));
```

```
}
```

```
function play() {
```

```
    // show pause button state
```

```
    if ($("#start").find(".glyphicon").hasClass("glyphicon-play")) {
```

```
        $("#start").find(".glyphicon").removeClass("glyphicon-play").addClass("glyphicon-pause");
```

```
    }
```

```
    // update params
```

```
    thingspeak['start'] = true;
```

```
    // start updates
```

```
    getUpdates();
```

```
    // check for new updates
```

```
    updates = setInterval('getUpdates()',5000);
```

```
    // save to local storage
```

```
    localStorage.setItem('thingspeak', JSON.stringify(thingspeak));
```

```
}
```

```
// toggle for channel info panel
```

```
function toggleChannelPanel(button) {
```

```
    // switch button state and show/hide panel
```

```
if ($(button).find(".glyphicon").hasClass("glyphicon-menu-down")) {  
  
    // call helper show function  
  
    showChannelPanel();  
  
    }  
  
    else {  
  
        // call helper hide function  
  
        hideChannelPanel();  
  
    }  
  
    }  
  
  
function showChannelPanel() {  
  
    // show minus button state  
  
    $('#toggle-channel-panel').find(".glyphicon").removeClass("glyphicon-menu-down").addClass("glyphicon-menu-up");  
  
    // show panel  
  
    $('#channel-panel').collapse('show');  
  
    }  
  
  
function hideChannelPanel() {  
  
    // show plus button state  
  
    $('#toggle-channel-panel').find(".glyphicon").removeClass("glyphicon-menu-up").addClass("glyphicon-menu-down");
```

```
// hide panel  
$('#channel-panel').collapse('hide');  
  
}  
  
function toggleConfigureChannelPanel() {  
  
// toggle channel panel configuration  
$('#channel-panel-config').collapse('toggle');  
  
}  
  
function showConfigureChannelPanel() {  
  
// show channel panel configuration  
$('#channel-panel-config').collapse('show');  
  
}  
  
function hideConfigureChannelPanel() {  
  
// hide channel panel configuration  
$('#channel-panel-config').collapse('hide');  
  
}  
  
</script>  
</head>  
<body>
```

```
<div class="container">

  <div class="panel panel-default">

    <div class="panel-heading">

      <div class="row">

        <div class="col-lg-6">

          <button type="button" class="btn btn-primary"
onClick="playPause(this)" id="start">

            <span class="glyphicon glyphicon-play"></span>

          </button>

        </div>

      </div>

    </div>

  <div class="panel-body">

    <h1 id="header"></h1>

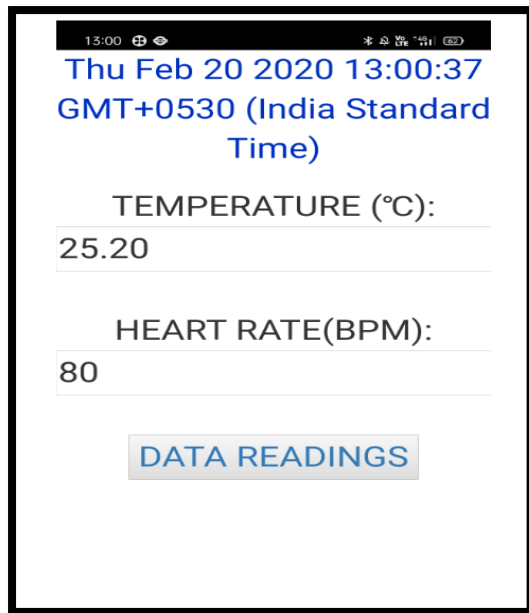
    <table class="table table-striped" id="output"></table>

  </div>

</div>

</body></html>
```





- **Code Efficiency:**

The code used for hardware was done on Arduino Ide under which we made use of supportable libraries for Nodemcu12E for which we need to installed the driver cp2102 and libraries of esp8266,dht11 and api key to send data.

The mobile app code s done in html and javascript as it gets data based on http get request and writing functions for it we use the url and api key to get the data in json format and display it in the screen as it a web page we converted combined zipped into an cordova apk using apache cordova app builder.

## 5.3 Testing approaches

- Testing Approach tells us how to test the entire project.
- There are two basic type of testing method:-
  - ❖ **Unit testing** where the basic modules are tested with the basic codes to check if the components are running properly individually.
  - ❖ **Integration testing** is where all the small single modules are connected together to perform a desired task; the final working of the project is tested from start to the end.

### 5.3.1 Unit Testing

Unit testing deals with testing a unit or module as a whole. This would test the interaction of many functions but, do confine the test within one module.

- **WEARABLE DEVICE**  
This module is working properly basically in this module the user or any individual wears the device and with connecting to the wifi or hotspot it gets internet connection and with the power supply given to it the device starts taking data through the sensors connected to it and measure heart rate and body temperature which is sent to the server which gets store over there
- **MOBILE APPLICATION**  
In this module the user himself or any family member or caretaker who is concerned with checking the health parameters can check the readings coming in this application as output of the data getting send by the wearable device .But only with proper authentication id and password one can get access to this data readings.

### **5.3.2 Integration Testing**

Brings all the modules together into a special testing environment, then checks for errors, bugs and interoperability. It deals with tests for the entire application. Application limits and features are tested here.

Here we tested our complete working of our hardware and software by running it together in which we faced few problems sometime data was displayed sometimes it took more time but after all tests done it worked properly. With proper internet connection and power supply the wearable works properly and the data stored in server gets displayed in the mobile application.

### **5.4 Modification and Improvements**

After testing we got to know that for any embedded projects to store data it's better to make use of cloud service and making use of any MQTT or any HTTP protocol for sending and receiving data. The mobile application needs supportable libraries for fetching data back from server which was leading us to errors in application development so we changed it and made use of HTTP GET request using HTML and JavaScript coding instead of Java which helped us getting the readings in mobile application.

## Chapter 6 Results and Discussion

### 6.1 Test Reports

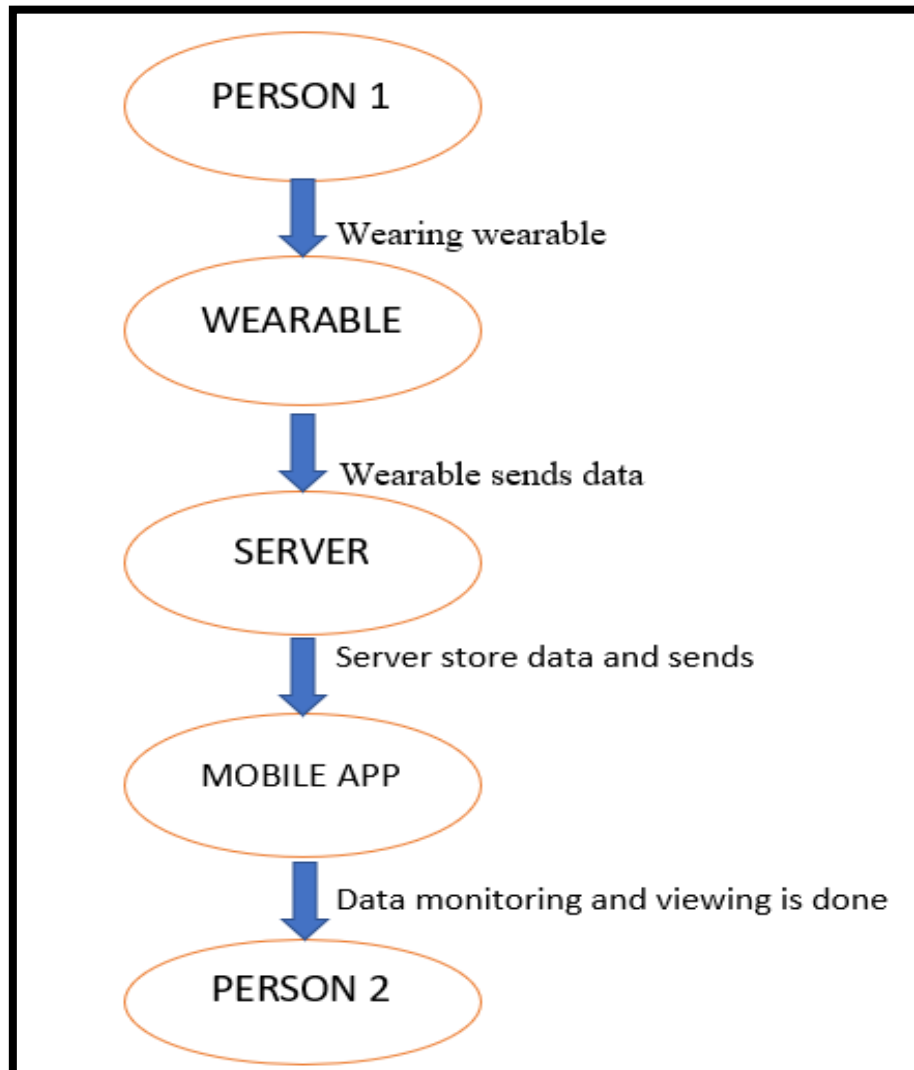
INPUT	OUTPUT
Wearable Device	Output as readings sent to the server properly.
Records	Data records coming from the device gets stored in the server properly.
View Records	After successful login the data readings are displayed in the mobile app properly.

**Table 6. 1 Test Reports**

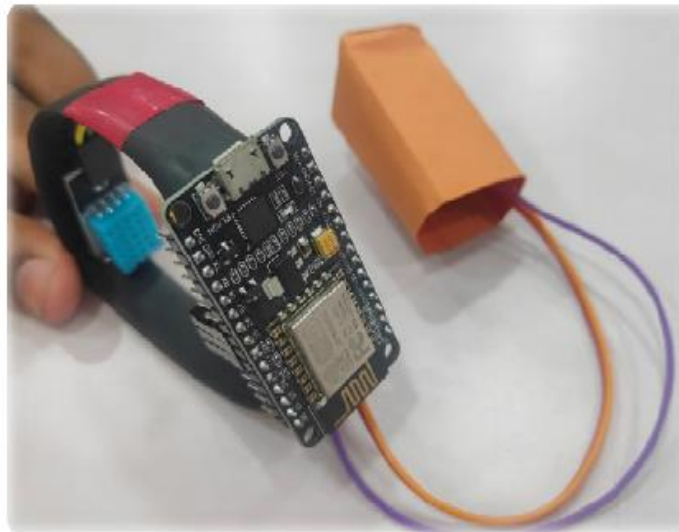
## 6.2 User Documentation

The project “LOW COST HEALTH MONITORING SYSTEM” is an Embedded solution that includes a light weight, portable wearable device to check heart rate and body temperature. The wearable device would continuously collect this data and send it with the help of a mobile app to the parents / caretakers.

The mobile app additionally can then be used to track the person’s heart rate and body temperature over a period of time.



1. Firstly the user or individual needs to wear the device and give power supply to it for the sensors to get data.



2. The data gets store in the server.
3. The mobile app can be accessed only with id given and the data can be seen over there



. Login Screen

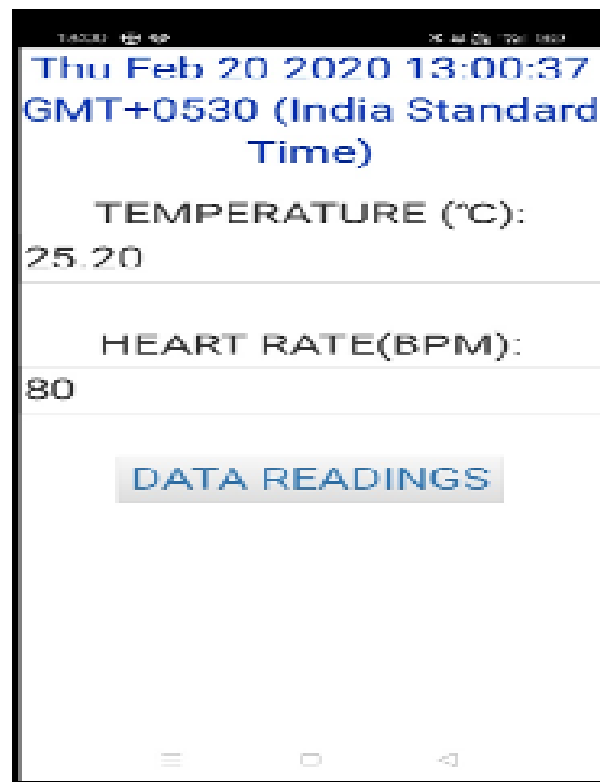
---

4. There will be a home page screen where there are options to select for displaying data



Home Screen

---



## Chapter 7 Conclusion

### 7.1 Conclusion

The system provides a real time solution of monitoring health of self or any family members with reliability. It is a very low cost and a possible way to record the health parameters of the person in a very proper way making it as a very helpful for peoples.

The previous data readings also facilitate tracking and monitoring health.

- LOW COST
- LIGHT WEIGHT
- PORTABLE

### 7.2 Limitation of the system

The system needs internet connectivity whenever used. The system needs to make use of proper power banks or power supply using any other sought of batteries may make the damage to the device or may result in heat up of the device. The device is currently recording two parameters of body temperature and heart rate. It can be added up with more parameters. For the time it can display records for a certain period of type approximately one hour it could be increased the time limit which is not implemented for the time being.

### 7.3 Future Scope of the Project

- The system can be made to track more parameters in addition to the current ones.
- Additionally, the GPS module can be used to identify the exact location of the person and if necessary timely help can be provided to the person.
- The application can be improved in a way to use the collected data in a much proper way by sending that data to either to doctors or the caretaker so they even they can understood the health about the person if needed.



## References

[https://github.com/Hanappy/pulse\\_sensor?files=1](https://github.com/Hanappy/pulse_sensor?files=1)

<https://ubidots.com/blog/connect-your-esp8266-to-any-available-wi-fi-network/>

<https://www.instructables.com/id/ESP8266-Communication-With-Server-and-ESP8266/>

<https://www.how2electronics.com/pulse-rate-monitoring-over-internet-using-thingspeak/>

<https://www.esp8266.com/viewtopic.php?f=32&t=10996>

<https://github.com/Isha2109/IOT-Heath-Monitoring-System/tree/Isha2109-patch-1?files=1>

<https://github.com/nothans/iot-debugger/tree/master/app>

## Bibliography

<https://github.com/hananabilabd/Reading-Sensor-Data-saving-it-to-ESP8266-Webserver-using-Xampp-Python-Visualization/blob/master/task0.ino>

<https://stackoverflow.com/questions/42135961/wificlient-notconnecting-to-xampp-server>

<https://alselectro.wordpress.com/2015/05/13/wi-fi-module-esp8266-3-connect-to-android-mobile/>

<https://github.com/>

<https://stackoverflow.com/>

<http://googlerobotic.com/androidfetching-data-thingspeak-arduino/>

<https://ieeexplore.ieee.org/document/8767280>

## Website Used

<https://thingspeak.com/>

<https://www.arduino.cc/>

<https://build.phonegap.com/>

## Glossary

**IDE:** An integrated development environment is a software application that provides comprehensive facilities to computer programmers for software development.

**GPIO:** A general-purpose input/output is an uncommitted digital signal pin on an integrated circuit or electronic circuit board whose behavior—including whether it acts as input or output—is controllable by the user at run time.

**HTTP:** HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.

**MQTT:** MQTT is an ISO standard publish-subscribe-based messaging protocol. It works on top of the TCP/IP protocol suite.

**API:** An application programming interface is an interface or communication protocol between a client and a server intended to simplify the building of client-side software.

**LAN:** A local area network is a computer network that interconnects computers within a limited area such as a residence, school, laboratory, university campus or office building.

## Summary

The overall state that we understood is health is wealth and having care for the loved ones is important as making time for them is too difficult. As our Project makes is a better way to have a proper monitoring over any individual health.

The way of using technology as a better way to have a proper handy and portable wearable to make it an easier job to check readings and making the job easy. There are many advanced wearables available but we made sure that it should be easy for people to use and they do not need to make any extra settings on their own which can be even use by common peoples.

## Further Reading

ESP8266: Get Started With ESP8266 Programming NodeMCU Using Arduino IDE  
by [Upskill Learning](#) (Author).

Internet of Things with ESP8266 by Marco Schwartz.

IOT Pulse sensor With Arduino UNO, Python and Thingspeak

By Guillermo Perez

# Plagiarism Report

## Results

Scan Properties

Number of Words : 959  
Results Found : 5

To or From

Binary Translator

To or From

PDF Converter

