**Chapter 4 – Implementation E-Triage health monitoring system using Telepresence**

**Index**

|  |  |
| --- | --- |
| Title | Page No. |
| Project Overview | 2 |
| Discussion about software and hardware requirements | 2 |
| Discussion about modules, components and their implementation | 5 |
| Wiring Details | 6 |
| Use Case Studies | 9 |
| Results with screenshots | 11 |
| Appendix – Pseudocode | 16 |

This project can be accessed at: <http://ptmanagement.ddns.net>

**4.1 Implementation**

The system is developed for monitoring the health condition of remote people with the use of body temperature (DS178B20), Heart rate and blood pressure (Sunrom Blood Pressure Sensor), and Blood Oxygen level (Pulse Oximeter). Raspberry Pi reads the sensor data and transmits them to the MySQL database for real-time monitoring of health condition. For detecting the severity of patient ESI (Emergency Severity Index) score will be calculated using sensor data with an ESI E-Triage algorithm and TRIAGE tag will be generated depends on the condition of a patient and CASE will be opened and assigned to one of the following depending on the condition of a patient: Emergency Doctor, General Practitioner or Physician. For real-time monitoring of the health condition of remote people, sensor data is displayed on the webpage can be seen either by Emergency Doctor, General Practitioner or Physician. In the telepresence, they can communicate with the patient using Audio, Video or Chat is implemented using WebRTC protocol and PHP WebSocket. They also can control the Mouse, Keyboard, and Screen of a Patient if the patient is not able to understand the sensor data.

**Following are modules:**

**4.1 Hardware Modules**

**- Raspberry Pi 3 model B+**

**- Sunrom Blood Pressure Sensor – Serial Output**

**- DS18B20** **Temperature Sensor**

**- CONTEC CMS50D Plus Fingertip Blood Oxygen Pulse Oximeter**

**- USB-TTL UART Module - CP2102**

**- USB Webcam, Mouse and Keyboard**

**4.1.1 Raspberry Pi 3 model B+**

The Raspberry Pi is a credit-card-sized computer that plugs into your TV and a keyboard. It is a capable little computer which can be used in electronics projects, and for many of the things that desktop PC does, like spreadsheets, word processing, browsing the internet, and playing games. It also plays high-definition video.

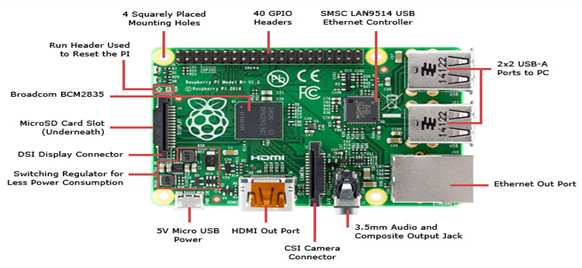
****

Figure 1: Raspberry Pi Pinout Diagram

**4.1.2 Sunrom Blood Pressure Sensor – Serial Output**

Blood Pressure & Pulse reading are shown on display with serial out for external projects of embedded circuit processing and display. Shows Systolic, Diastolic and Pulse Readings. Compact design fits over your wrist like a watch. Easy to use wrist style eliminates pumping.



Figure 2: Sunrom Blood Pressure Sensor – Serial Output

**4.1.3 Temperature Sensor - DS18B20**

Temperature sensor is a device which is designed specifically to measure the hotness or coldness of an object. DS18B20 is a precision IC temperature sensor with its output proportional to the temperature (in °C). With DS18B20, the temperature can be measured more accurately than with a thermistor. It also possesses low self-heating and does not cause more than 0.1 °C temperature rise in still air. The operating temperature range is from -55°C to 125°C.The DS18B20’s low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy.

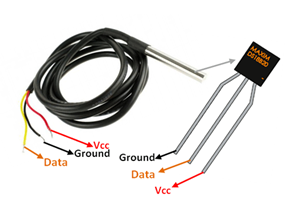
****

Figure 3: DS18B20 Temperature Sensor

**4.1.4 CONTEC CMS50D Plus Fingertip Blood Oxygen Pulse Oximeter**

CMS50D+ Pulse Oximeter is a portable equipment adopting advanced technology, it mainly checks SpO2 and PR value through the finger, which provides advanced, non-invasive and scientific means for quantitative measurement of oxygen saturation.

****

Figure 4: CMS50D+ Oximeter

**4.1.5 USB-TTL UART Module - CP2102**

This is an USB2.0 to TTL UART Converter module which is based on CP2102 Bridge by SiLabs. This module can be used with Laptop's which don't have standard serial port.

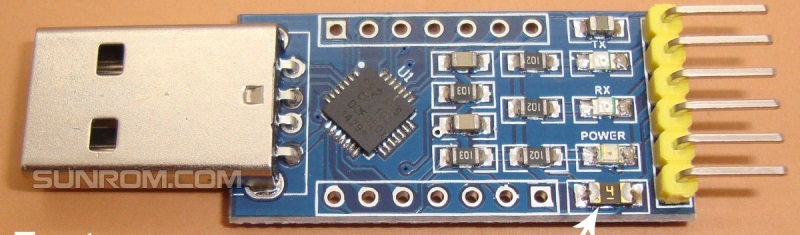
****

Figure 5: USB-TTL UART Module - CP2102

**4.1.5 USB Webcam**

A USB webcam is a camera that connects to a Raspberry PI through plugging it in to a USB port on the machine. The video is fed to the computer where a software application lets you view the pictures and transfer them to the Internet.

**4.2 Software Modules**

**- Raspbian OS**

**- Python IDLE**

**- Apache Web Server**

**- Let's Encrypt - Free SSL/TLS Certificates**

**- Web Development Tools & Technology (PHP, HTML, CSS & MySQL)**

**- Tornado Python**

**4.2.1 Raspbian OS**

Raspbian is a Debian-based computer operating system for Raspberry Pi. Raspbian comes pre-installed Python and other development tools and integrated development environments.

**4.2.2 Python IDLE**

IDLE is an integrated development environment for Python.

**4.2.3 Apache Web Server**

Apache is used in the development of this project. It is also the most commonly used Web server on Linux systems. Web servers are used to serve Web pages requested by client computers. Clients typically request and view Web pages using Web browser applications such as Firefox, Opera, Chromium, or Internet Explorer.

**4.2.4 Let's Encrypt - Free SSL/TLS Certificates**

Let's Encrypt is a certificate authority that provides X.509 certificates for Transport Layer Security (TLS) encryption at no charge. To work with audio and video transmission, SSL needs to be enabled.

**4.2.5 Web Development Tools & Technology (PHP, HTML, CSS & MySQL)**

A LAMP Stack is a set of open-source software that can be used to create websites and web applications. PHP, HTML, CSS & MySQL are used to develop this project.

**4.2.6 Tornado Python**

Tornado is a Python web framework and asynchronous networking library. In this project, tornado is used to enable telepresence. Medical Team can view patient’s computer remotely.

## Software Modules Installation

## I installed a webserver on Raspberry Pi which includes Apache2 + PHP + MYSQL. The steps which I followed to install a webserver on raspberry Pi is as follow:

## sudo apt-get install apache2

## sudo apt-get install mysql-server

## sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql

## Once the installation is finished, I tested the apache 2 server is working properly by opening. <http://localhost/> It shows you apache welcome page If it is successfully installed.

## Following step is to install PHP. PHP is an interpreted language and in case of using the server in our system, we required PHP for server side scripting. According to our fourth step of the project -Dynamic Data, PHP is the best option for Dynamic Data because PHP is mainly used to make a site dynamic where the user sends information to the server which returns the modified result according to the information.

## To check the installation of PHP I created hello.php file with following code.

## <?php echo phpinfo(); ?>

## After creating and storing the file at the server location, open the url: <http://localhost/> will redirect you to the php information page. If you got redirected to php info page, PHP is installed successfully.

## Another step is to install MYSQL for storing the vital information to the database for future work.

## To verify the MYSQL is working correctly or not, you need to connect with mysql by executing below command in terminal again.

## sudo mysql --user=root

## If it gets logged In to the server, you successfully connectly to MYSQL and it is working fine.

## Last step is to install SSL certificates. This is used to provide secure connection between host and server. In this project SSL is used to provide enable audio chat and video chat. As google chrome removed their support for unsecured connection. Thus, SSL is required to work with audio and video APIs.

## I have used Let’s Encrypt SSL Certificates. Let's Encrypt is a free, automated, and open certificate authority by the non-profit Internet Security Research Group (ISRG) and provides free certificates to researcher and students. The installation is fairly easy. Following commands need to be executed.

## wget <https://dl.eff.org/certbot-auto>

## chmod a+x certbot-auto

## sudo ./path/to/certbot-auto –apache

## Running this command will get a certificate for you and have Certbot edit your Apache configuration automatically to serve it.

## Following are module implementation

## 4.2. Implementation of reading body temperature, Blood Oxygen Level, Blood Pressure and Heart Rate.

Python has been used for reading and processing data from the sensor and sends vitals information to php. The code is executed every second and if vital information is available from the patients. The E-Triage algorithm calculates the abnormal vitals and opens a case which sends alerts to general practitioner or emergency team.

## 4.2.1 Body Temperature Sensor Module

## Component Used to connect with Raspberry PI.

## 1. DS18B20 sensor

## 2. Breadboard

## 3. Male to Male jumper cables.

## DS18B20 sensor is used to measure body temperature.

## I:\My Drive\images\Raspberry-Pi-DS18B20.png

## Figure 6: Wiring to output the Temperature Diagram

## The DS18B20 has three separate pins for ground, data, and Vcc:

## I:\My Drive\images\DS18B20 Pinout-Diagram.png

## Figure 7: Pinout Diagram

1. Connect Sensor as shown on Figure 6 above.
2. Load 1-wire kernel modules that come pre-installed but not loaded: sudo modprobe w1-gpio and then sudo modprobe w1\_therm
3. Add lines w1-gpio and w1\_therm into /etc/modules using sudo nano /etc/modules so they get loaded automatically the next time you restart it.
4. Find your sensor: ls /sys/bus/w1/devices/ it should look like 28-00000393268a

## Oxygen Level Sensor Module

## CONTEC CMS50D Plus Fingertip Blood Oxygen Pulse Oximeter is used to measure Blood Oxygen level.

## Component Used to connect with Raspberry PI.

## 1. CONTEC CMS50D+ Sensor

## 2. USB Cable

## 3. AAA Batteries

## I:\My Drive\images\fritzing oxygen.png

## Figure 8: Wiring Oxygen Level Sensor

This sensor is connected to Raspberry PI via USB port. /dev/usbTTY0 will be available immediately to fetch data. It does not start the streaming as soon as it is connected. Streaming is activated by sending series of hex data(\x7d\x81\xa1\x80\x80\x80\x80\x80\x80).

The patient needs to insert a finger into the sensor to start measuring the blood oxygen level.

## Blood Pressure and Heart Rate Monitor Sensor Module

## Component Used to connect with Raspberry PI.

## 1. Blood Pressure Sensor - Serial output Sensor

## 2. USB-TTL UART Module - CP2102

## I:\My Drive\images\pcb.png

## Figure 9: Wiring Blood Pressure Sensor

## I have used Blood Pressure Sensor - Serial output by Sunrom Technologies. It provides Blood Pressure & Pulse reading are shown on display with serial out for external projects of embedded circuit processing and display. Shows Systolic, Diastolic and Pulse Readings. It has Easy to use wrist style eliminates pumping.

## Once the patient has attached blood pressure sensor to their wrist, they can start measuring by clicking on the start button on the device. Once the blood pressure and heart rate are measured, the data will be displayed on to the webpage.

* 1. **Module for Video, Audio and Text chat WebRTC Communication Using RatchetSocket**

Video, Audio and Text chat are used to enable telepresence.

**WebRTC** is an open source project to enable realtime communication of audio, video and data in Web and native apps.

WebRTC has several JavaScript APIs

getUserMedia(): capture audio and video.

MediaRecorder: record audio and video.

RTCPeerConnection: stream audio and video between users.

RTCDataChannel: stream data between users.

**PHP WebSocket:** WebSockets are a bi-directional, full-duplex, persistent connection from a web browser to a server. Once a WebSocket connection is established the connection stays open until the client or server decides to close this connection. With this open connection, the client or server can send a message at any given time to the other. This makes web programming entirely event driven, not (just) user initiated. It is stateful. As well, at this time, a single running server application is aware of all connections, allowing you to communicate with any number of open connections at any given time.

Ratchet is a loosely coupled PHP library providing developers with tools to create real time, bi-directional applications between clients and servers over WebSocket.

For the audio and video chat to work, website needs to be secured (SSL Installed).

* 1. **Module for screen sharing**

To work remotely with the patient’s screen in the event of patient is not able to understand the working of the system or not able to measure vital information correctly.

Medical team can take the access of the patients’ screen remotely and can access the mouse and keyboard.

To develop this module, Tornado is used which is a Python web framework. Application developed in tornado captures the remote patients’ screen every second and convert it into base64 encoding and stream it online with the use of WebSocket.

Medical team can remotely view patients’ screen by accessing URL.

## 4.5. E-Triage Algorithm

## E-Triage Algorithm is the main part of the project and developed in JavaScript.

## E-Triage Algorithm calculates Triage code based on users’ vital information and generates a case if abnormal information found. For example, if heart rate is greater than 140. It opens a case and sends alerts to emergency team along with the rest of the vital information. It categorizes patients’ into following categories according to their triage value.

|  |  |
| --- | --- |
| Notifier - Emergency Team1. Triage Code - Immediate resuscitationHyperthermic (>=100.5 °F)Severe hypoxia (SpO2 <=89) | Notifier – Emergency Team2. Triage Code - EmergencyHypothermia (<=94.0 °F)Severe bradycardia (<=49 bpm)Hypotension (<=99 mmHg)Moderate hypoxia (SpO2 90–94) |
| Notifier - General Practitioner3. Triage Code - UrgentMild hypothermia (94.1–96.2 °F)Mild hyperthermia (99.3–100.4 °F)Severe tachycardia (>=130 bpm)High tachycardia (120–129 bpm)Mild hypotension (100–107 mmHg) | Notifier - General Practitioner4. Triage Code - Semi UrgentMild tachycardia (105–109 bpm)Moderate tachycardia (110–119 bpm)Mild hypertension (177–199 mmHg)Hypertension (>=200 mmHg) |

## E-Triage Algorithm Is calculated using JavaScript.

## Once E-Triage code is assigned and abnormalities found. It sends the information to Emergency Team and General Practitioner accordingly. The patients’ information will be stored into database. The following code is used to detect abnormalities and storing information to Database.

A case will be opened and Case ID and E-Triage code is assigned which can be seen by either Emergency Team or General Practitioner.

**4.6. Case Studies**

**Actors:**

The Actors of the system are Patient, Guardian and Doctor.

**Tools related to use case diagram**

Web application comes with attached sensors that can measure vitals data.

Patients will be able to know the results immediately once the vitals are measured weather he/she has any immediate abnormal vitals.

Patient’s data will be sent to either the general practitioner or emergency team based on the urgency of the case.

General Practitioner or Emergency Team can communicate with the patient with the easy to use audio video and chat built in the system enabling telepresence.

General Practitioner or Emergency Team can view the patients’ screen enabling telepresence.

General Practitioner or Emergency Team can view the history of past patients’ vital information.

Reduces the time to visit the health center.

Emergency Team receives alerts to intervene if the patient is at increased risk for urgent care or hospital readmission.

**Use Case Scenario**

**Scenario 1:**

Let’s say this system is implemented in the rural area which has very few hospitals and with minimal resources, for Example, No Ambulance Services, very few health personals.

One person is living in a rural area and has a high body temperature of 110F. Instead of visiting hospitals or calling an ambulance. He uses this E-Triage system and by just measuring body temperature. This E-Triage system will open a case, and the Emergency Team is notified which can further assist the patient with the inbuilt audio, video, and chat.

This reduces the response time that patients’ need to communicate with the Medical Team. This doesn’t involve calling 911 or health care system. Emergency Team will have immediate access to patients’ data, and Emergency Team can also view the patient with the use of inbuilt video chat.

The Emergency Team can further assist the patient using Audio, Video, and chat.

**Scenario 2:**

Another older adult getting medical treatment at home and not able to do regular movement even as simple as walking. If the body sensors are attached, the general practitioner will be able to see every second detail of the patient and can communicate.

**Use cases:**

I have identified a set of use cases based on the functionalities and goals of the application.

• Login: This use case denotes a set of actions required for Subject to login into the application.

• View Patient’s Live Vitals: This use case denotes a set of actions required by General Practitioner or Emergency Team to read live vital information.

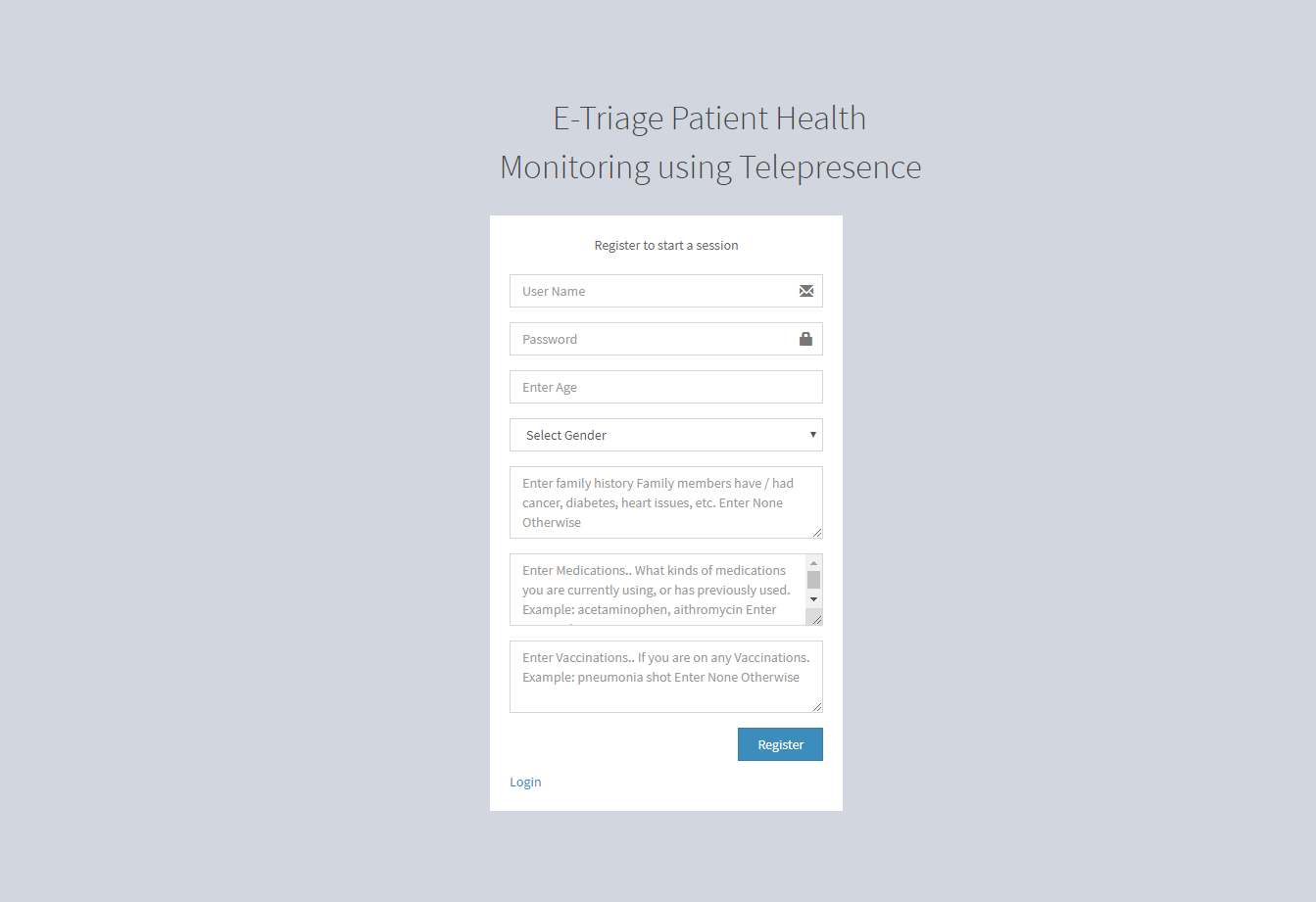
• View History of Patient’s Vitals: This use case denotes a set of actions required by General Practitioner or Emergency Team to read history of Patient’s Vital Information.

• Communicate with Patient: This use case denotes a set of actions required by General Practitioner or Emergency Team to Communication with Patient Using Audio, Video and Text.

• Patient’s Screen Sharing: This use case denotes a set of actions required by General Practitioner or Emergency Team to view user’s screen remotely.

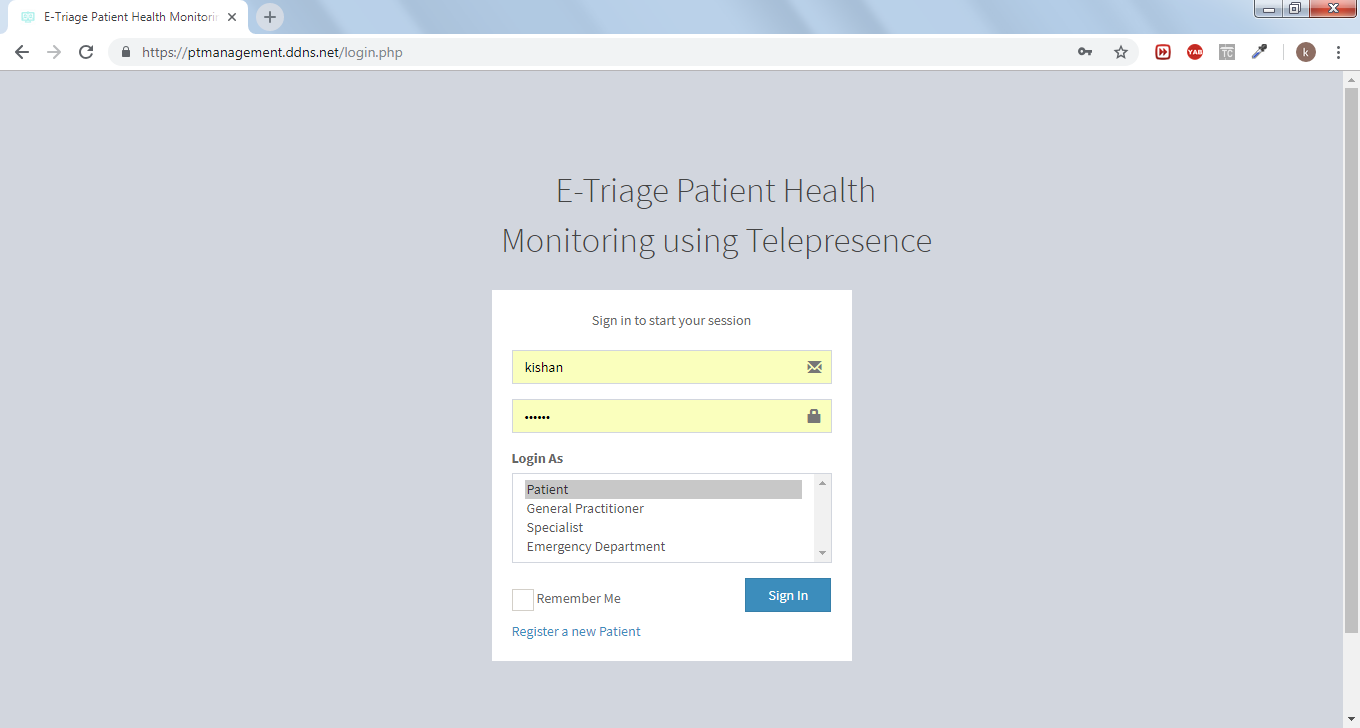
**4.8 Results with screenshot:**

1. **Registration Page**

****

The new user can enroll in this system after creating their account from the registration page. They need to create an account by entering username, password, age, gender, family history, medication and vaccination. Family history, medication and vaccination details will be displayed to the Emergency Team and General Practitioner to fasten up the treatment process.

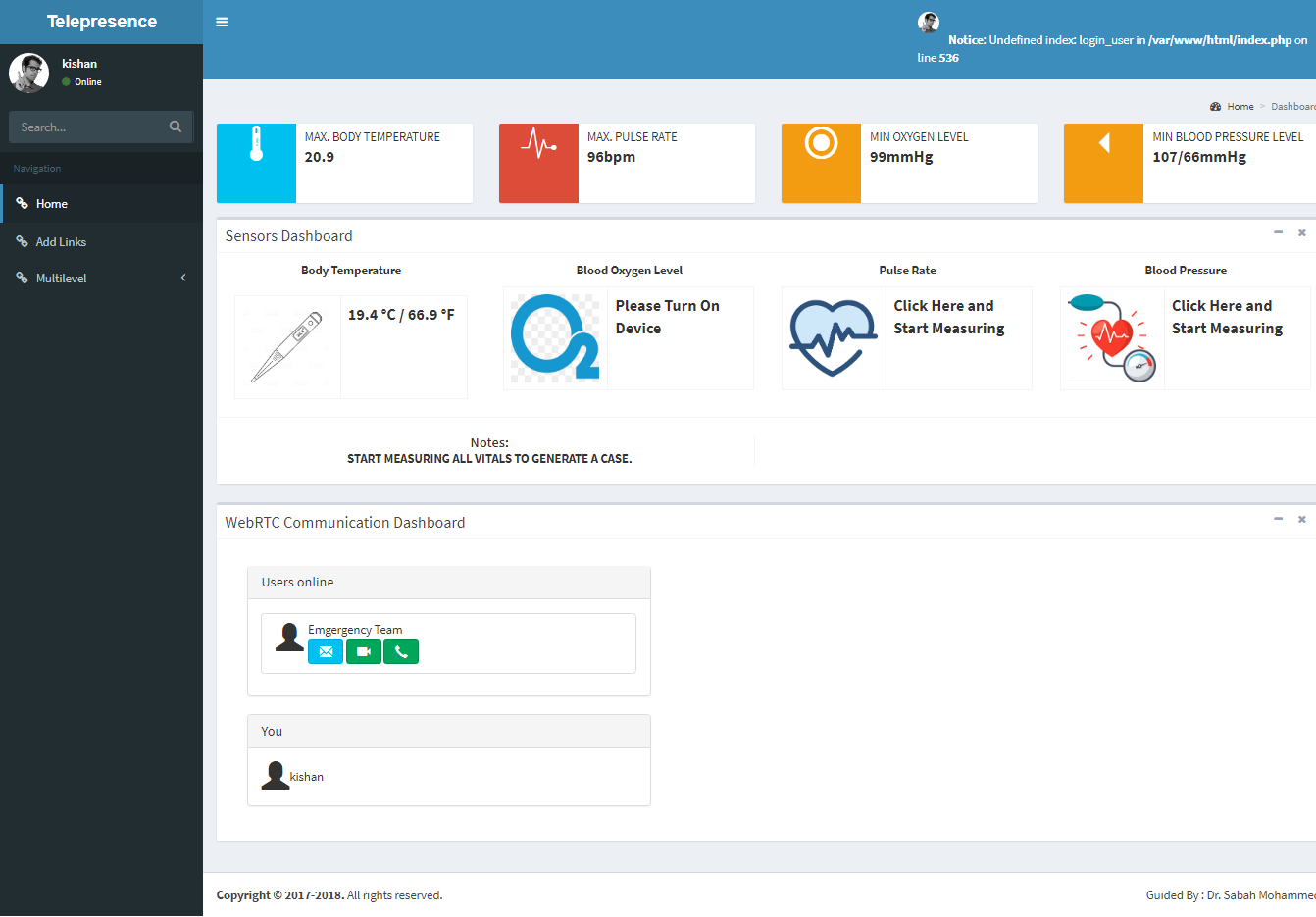
1. **Login Page**

****

Emergency Team, General Practitioner, Specialist, and Patient enter their unique credentials. Once the credentials are verified, the login page will be navigated to their related webpages, where they can monitor the patient’s vital information.

Here credentials must be kept confidential by the doctor and caretaker to protect the privacy of the patient data.

1. **Patient’s Dashboard**

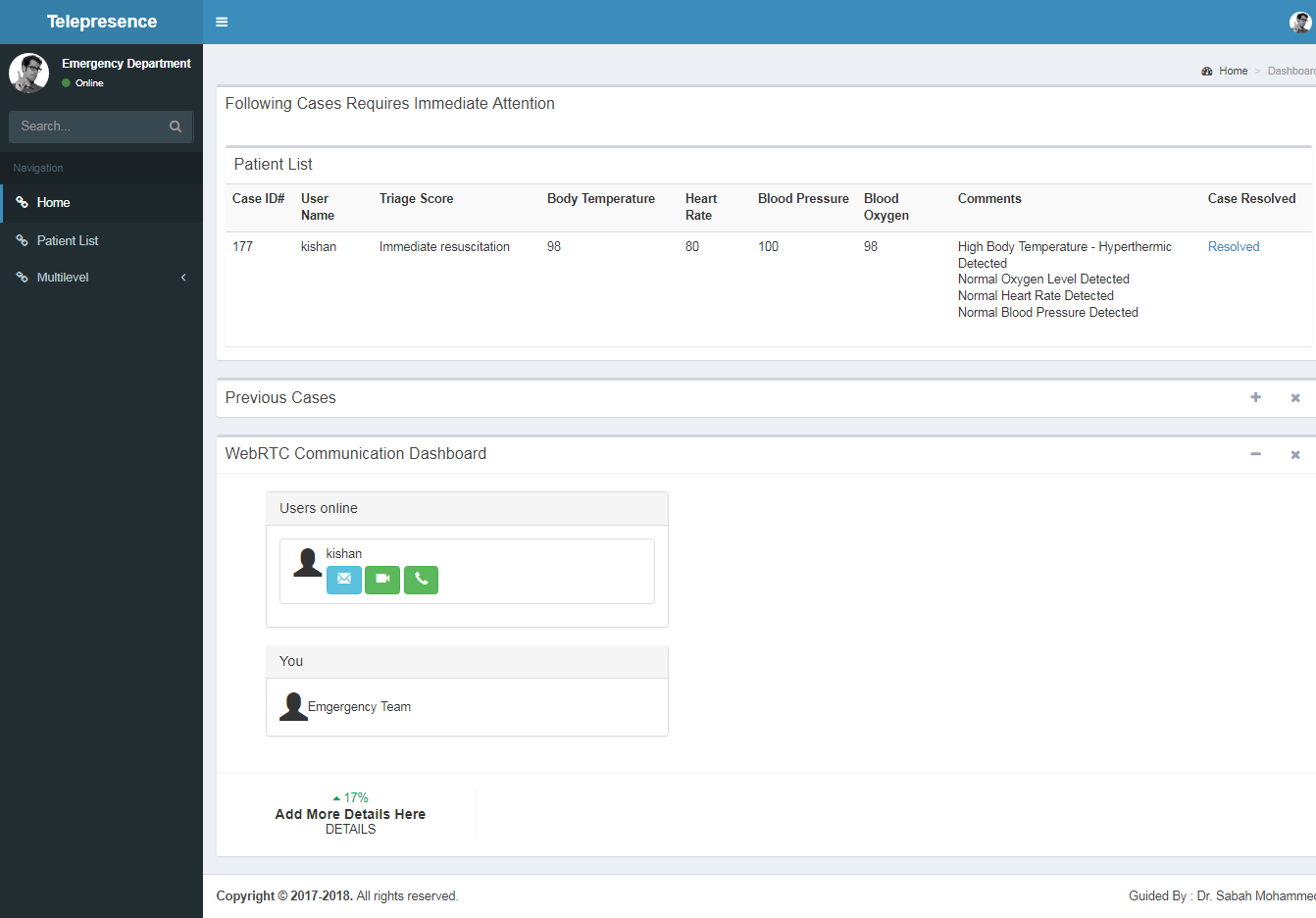


Patient’s Dashboard displays maximum body temperature, maximum pulse rate, minimum oxygen level and minimum blood pressure level from the past data.

Once the patient has measured vital data. It will be displayed here automatically without refreshing the page.

Patient has also an option for talking with the Emergency Team if there are abnormalities in patient’s vitals.

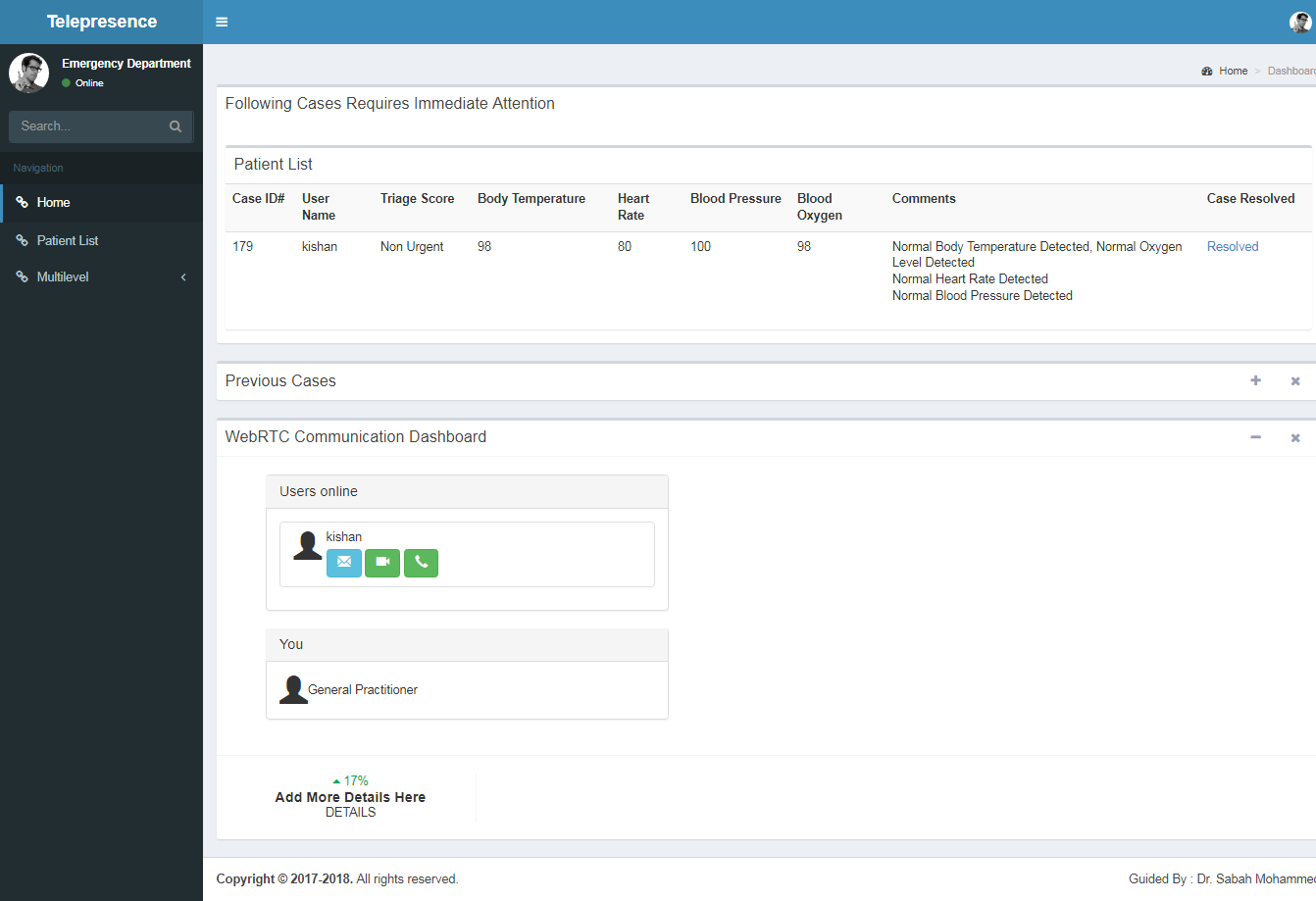
1. **Emergency Team Dashboard**

****

Emergency Team’s Dashboard shows the cases where immediate attention is required with the comment option which shows the condition of the patient and Triage Score which shows the urgency of the patient. It also shows Case ID, Patient’s Name, body temperature, heart rate, oxygen level and blood pressure details.

Emergency Team has also an option for talking with the patient to quickly check the condition of the patient.

1. **General practitioner Dashboard**

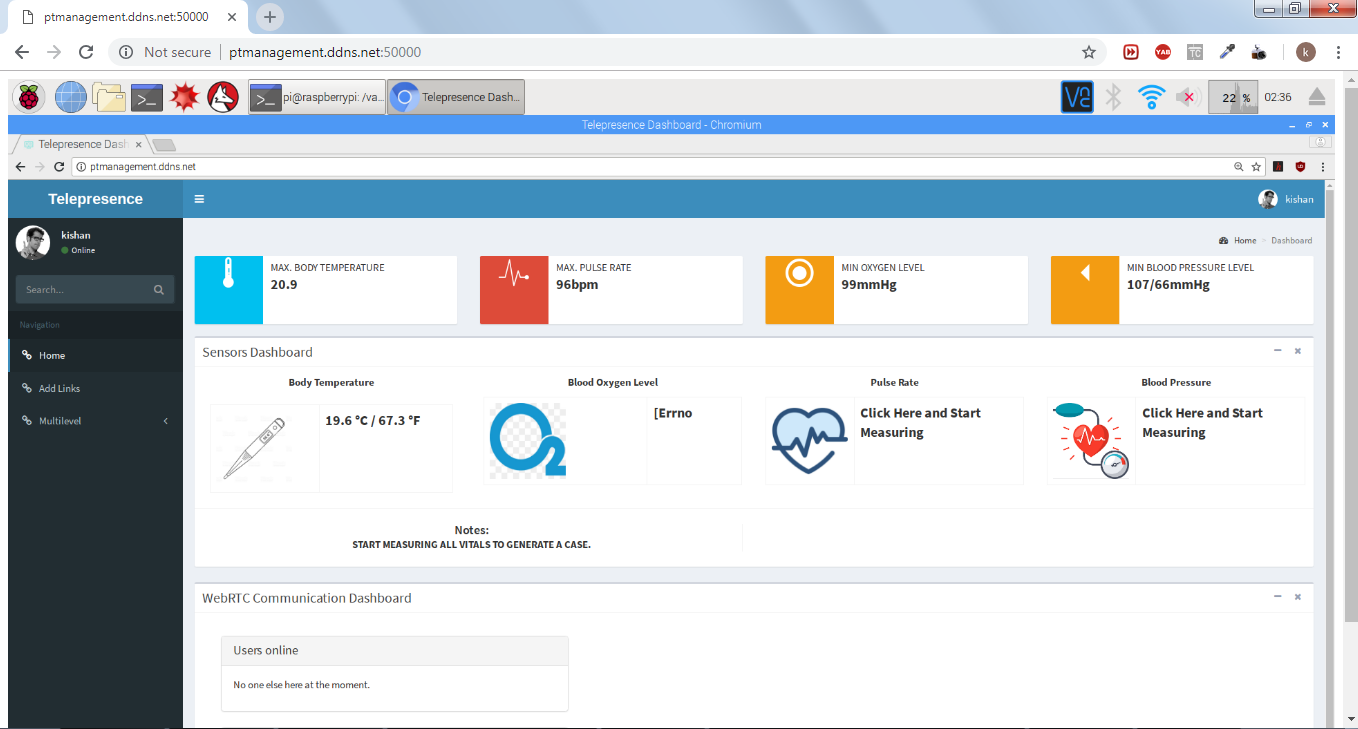


General Practitioner’s Dashboard shows the cases where patient condition which do not require immediate attention with the comment option which shows the condition of the patient and Triage Score which shows the urgency of the patient. It also shows Case ID, Patient’s Name, body temperature, heart rate, oxygen level and blood pressure details.

General Practitioner has also an option for talking with the patient to quickly check the condition of the patient.

1. **Remote Patient’s Screen Sharing**

General Practitioner and Emergency Team have a provision on the dashboard to click on the URL which will open another tab where they can view patients screen. This option does not require downloading any software or plugin. General Practitioner or Emergency Team can also control the mouse and keyboard of the remote system.

****

**4.9 Appendix – Pseudocode**

## To determine E-Triage for body temperature

## 

## To determine E-Triage for Blood Oxygen Level

## 

## To determine E-Triage for Blood Pressure Level

## 

## To determine E-Triage for Heart Rate

## 

## Once E-Triage code is assigned and abnormalities found. It sends the information to Emergency Team and General Practitioner accordingly. The patients’ information will be stored into database. The following code is used to detect abnormalities and storing information to Database.

<?php

**include** './config.php';

**if** ($link->connect\_error) {

**die**("Connection failed: " . $link->connect\_error);

}

$bodytemperature = (**int**) $\_REQUEST['bodytemperature'];

$oxygenlevel = (**int**) $\_REQUEST['oxygenlevel'];

$bloodpressure1 = (**int**) $\_REQUEST['bloodpressure1'];

$heartrate = (**int**) $\_REQUEST['heartrate'];

$triagescore = (**int**) $\_REQUEST['TriageScore'];

$notifier = **array**();

$triage\_score = **array**();

**if** ($bodytemperature != 0) {

**if** ($bodytemperature >= 100.5) {

**array\_push**($notifier, 4);

**array\_push**($triage\_score, 1);

$TriageScore = 'Immediate resuscitation';

// echo 'notified to emergency';

} **else** **if** ($bodytemperature > 97.2 && $bodytemperature < 99.3) {

**array\_push**($notifier, 2);

**array\_push**($triage\_score, 5);

$TriageScore = 'Non Urgent';

// var\_dump($notifier);

// echo 'body temperature';

// echo 'notified to general';

}

}

//echo 'oxygen level is' . $oxygenlevel;

**if** ($oxygenlevel != 0) {

**if** ($oxygenlevel <= 89) {

**array\_push**($notifier, 4);

**array\_push**($triage\_score, 1);

$TriageScore = 'Immediate resuscitation';

// echo 'notified to emergency';

} **else** **if** ($oxygenlevel >= 90 && $oxygenlevel <= 94) {

**array\_push**($notifier, 4);

**array\_push**($triage\_score, 2);

$TriageScore = 'Emergency';

// echo 'notified to emergency';

} **else** **if** ($oxygenlevel > 94) {

**array\_push**($notifier, 2);

$TriageScore = 'Non Urgent';

**array\_push**($triage\_score, 5);

// echo 'oxygen level';

// echo 'notified to general';

}

}

**if** ($bloodpressure1 != 0) {

**if** ($bloodpressure1 <= 120) {

**array\_push**($notifier, 2);

$TriageScore = 'Non Urgent';

**array\_push**($triage\_score, 5);

// echo 'bloodpressure';

// echo 'notified to general';

} **else** **if** ($bloodpressure1 > 120 && $bloodpressure1 <= 129) {

**array\_push**($notifier, 2);

$TriageScore = 'Semi Urgent';

**array\_push**($triage\_score, 4);

// echo 'notified to general';

} **else** **if** ($bloodpressure1 >= 130 && $bloodpressure1 <= 139) {

**array\_push**($notifier, 2);

$TriageScore = 'Urgent';

**array\_push**($triage\_score, 3);

// echo 'notified to general';

} **else** **if** ($bloodpressure1 >= 140) {

**array\_push**($notifier, 4);

$TriageScore = 'Emergency';

**array\_push**($triage\_score, 2);

**echo** 'heart rate is executing';

// echo 'notified to emergency';

}

}

**if** ($heartrate != 0) {

**if** ($heartrate >= 60 && $heartrate <= 100) {

**array\_push**($notifier, 2);

$TriageScore = 'Non Urgent';

**array\_push**($triage\_score, 5);

} **else** **if** ($heartrate < 60 && $heartrate > 40) {

**array\_push**($notifier, 2);

$TriageScore = 'Semi Urgent';

**array\_push**($triage\_score, 4);

// echo 'notified to general';

} **else** **if** ($heartrate <= 40) {

**array\_push**($notifier, 4);

$TriageScore = 'Emergency';

// echo 'notified to emergency';

**array\_push**($triage\_score, 2);

} **else** **if** ($heartrate > 100) {

**array\_push**($notifier, 4);

$TriageScore = 'Emergency';

// echo 'notified to emergency';

**array\_push**($triage\_score, 2);

}

}

$notifiermain = **max**($notifier);

$notifier = $notifiermain;

$triagescoremain = **min**($triage\_score);

$triage\_score = $triagescoremain;

**if** ($triage\_score == 1) {

$triagescore = 'Immediate resuscitation';

} **else** **if** ($triage\_score == 2) {

$triagescore = 'Emergency';

} **else** **if** ($triage\_score == 3) {

$triagescore = 'Urgent';

} **else** **if** ($triage\_score == 4) {

$triagescore = 'Semi Urgent';

} **else** **if** ($triage\_score == 5) {

$triagescore = 'Non Urgent';

}

**if** ($\_REQUEST['bodytemperature']) {

$timestamp = $\_REQUEST['timestamp'];

$userid = $\_REQUEST['userid'];

$casetype = $\_REQUEST['casetype'];

$comments = $\_REQUEST['comments'];

$sql = "select \* from cases where userid = **$userid** and casetype = 'new'";

$sqlresult = $link->query($sql);

$row = $sqlresult->fetch\_assoc();

**if** ($row) {

**if** (**strcmp**($comments, $row['Comments']) != 0) {

$comments = $comments . ", <br>" . $row['Comments'];

$sql = "update cases set temperature = **$bodytemperature**, Comments = '**$comments**' where userid = **$userid** and casetype = 'new'";

$link->query($sql);

} **else** {

$sql = "update cases set temperature = **$bodytemperature** where userid = **$userid** and casetype = 'new'";

$link->query($sql);

}

} **else** {

$sql = "insert into cases(id, temperature, timestamp, userid, notifier, casetype, TriageScore, Comments) values('', **$bodytemperature**, '**$timestamp**', **$userid**, **$notifier**, '**$casetype**', '**$triagescore**', '**$comments**')";

$link->query($sql);

}

}

//For oxygenlevel

**if** ($\_REQUEST['oxygenlevel']) {

$timestamp = $\_REQUEST['timestamp'];

$userid = $\_REQUEST['userid'];

$casetype = $\_REQUEST['casetype'];

$comments = $\_REQUEST['comments'];

$sql = "select \* from cases where userid = **$userid** and casetype = 'new'";

$sqlresult = $link->query($sql);

$row = $sqlresult->fetch\_assoc();

**if** ($row) {

**if** (**strcmp**($comments, $row['Comments']) != 0) {

$comments = $comments . ", <br>" . $row['Comments'];

$sql = "update cases set bloodoxygen = '**$oxygenlevel**', Comments = '**$comments**' where userid = **$userid** and casetype = 'new'";

$link->query($sql);

} **else** {

$sql = "update cases set bloodoxygen = **$oxygenlevel** where userid = **$userid** and casetype = 'new'";

$link->query($sql);

}

} **else** {

$sql = "insert into cases(id, bloodoxygen, timestamp, userid, notifier, casetype, TriageScore, Comments) values('', **$oxygenlevel**, '**$timestamp**', **$userid**, **$notifier**, '**$casetype**', '**$triagescore**', '**$comments**')";

$link->query($sql);

}

}

//For oxygenlevel

**if** ($\_REQUEST['bloodpressure1']) {

$bloodpressure1 = $\_REQUEST['bloodpressure1'];

$bloodpressure2 = $\_REQUEST['bloodpressure1'];

$bloodpressure = $bloodpressure + "/" + $bloodpressure2;

$timestamp = $\_REQUEST['timestamp'];

$userid = $\_REQUEST['userid'];

$casetype = $\_REQUEST['casetype'];

$comments = $\_REQUEST['comments'];

$sql = "select \* from cases where userid = **$userid** and casetype = 'new'";

$sqlresult = $link->query($sql);

$row = $sqlresult->fetch\_assoc();

**if** ($row) {

**if** (**strcmp**($comments, $row['Comments']) != 0) {

$comments = $comments . ", <br>" . $row['Comments'];

$sql = "update cases set bloodpressure = '**$bloodpressure**', Comments = '**$comments**' where userid = **$userid** and casetype = 'new'";

$link->query($sql);

} **else** {

$sql = "update cases set bloodpressure = **$bloodpressure** where userid = **$userid** and casetype = 'new'";

$link->query($sql);

}

} **else** {

$sql = "insert into cases(id, bloodpressure, timestamp, userid, notifier, casetype, TriageScore, Comments) values('', '**$bloodpressure**', '**$timestamp**', **$userid**, **$notifier**, '**$casetype**', '**$triagescore**', '**$comments**')";

$link->query($sql);

}

//$link->query($sql);

}

//For Heart Rate

**if** ($\_REQUEST['heartrate']) {

$timestamp = $\_REQUEST['timestamp'];

$userid = $\_REQUEST['userid'];

$casetype = $\_REQUEST['casetype'];

$comments = $\_REQUEST['comments'];

$sql = "select \* from cases where userid = **$userid** and casetype = 'new'";

$sqlresult = $link->query($sql);

$row = $sqlresult->fetch\_assoc();

**if** ($row) {

**if** (**strcmp**($comments, $row['Comments']) != 0) {

$comments = $comments . ", <br>" . $row['Comments'];

// echo 'inserting comments';

$sql = "update cases set heartrate = **$heartrate**, Comments = '**$comments**' where userid = **$userid** and casetype = 'new'";

$link->query($sql);

} **else** {

// echo 'not inserting comments';

$sql = "update cases set heartrate = **$heartrate** where userid = **$userid** and casetype = 'new'";

$link->query($sql);

}

} **else** {

// echo 'not inserting comments';

$sql = "insert into cases(id, heartrate, timestamp, userid, notifier, casetype, TriageScore, Comments) values('', **$heartrate**, '**$timestamp**', **$userid**, **$notifier**, '**$casetype**', '**$triagescore**', '**$comments**')";

$link->query($sql);

}

}

$userid = $\_REQUEST['userid'];

//update traiage score if database value is different from calculated triage score.

$sql = "select \* from cases where userid = **$userid** and casetype = 'new'";

$sqlresult = $link->query($sql);

$row = $sqlresult->fetch\_assoc();

$triagescore = $row['TriageScore'];

//$triage\_score

**if** ($triagescore == 'Immediate resuscitation') {

$triagescore = 1;

} **else** **if** ($triagescore == 'Emergency') {

$triagescore = 2;

} **else** **if** ($triagescore == 'Urgent') {

$triagescore = 3;

} **else** **if** ($triagescore == 'Semi Urgent') {

$triagescore = 4;

} **else** **if** ($triagescore == 'Non Urgent') {

$triagescore = 5;

}

**if** ($triagescore > $triagescoremain) {

**if** ($triagescoremain == 1) {

$triagescoremain = 'Immediate resuscitation';

} **else** **if** ($triagescoremain == 2) {

$triagescoremain = 'Emergency';

} **else** **if** ($triagescoremain == 3) {

$triagescoremain = 'Urgent';

} **else** **if** ($triagescoremain == 4) {

$triagescoremain = 'Semi Urgent';

} **else** **if** ($triagescoremain == 5) {

$triagescoremain = 'Non Urgent';

}

$sql = "update cases set TriageScore='**$triagescoremain**' where userid = **$userid** and casetype = 'new'";

$link->query($sql);

$sql = "select notifier from cases where userid = **$userid** and casetype = 'new'";

$sqlresult = $link->query($sql);

$row = $sqlresult->fetch\_assoc();

**if** ($row['notifier'] < $notifiermain) {

$sql = "update cases set notifier=**$notifiermain** where userid = **$userid** and casetype = 'new'";

$link->query($sql);

// echo $sql;

}

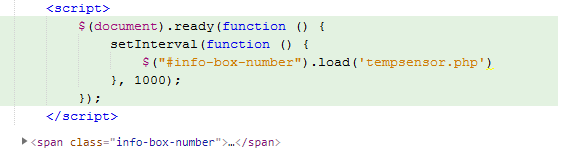
?>

A case will be opened and Case ID and E-Triage code is assigned which can be seen by either Emergency Team or General Practitioner.

Code for the body temperature sensor



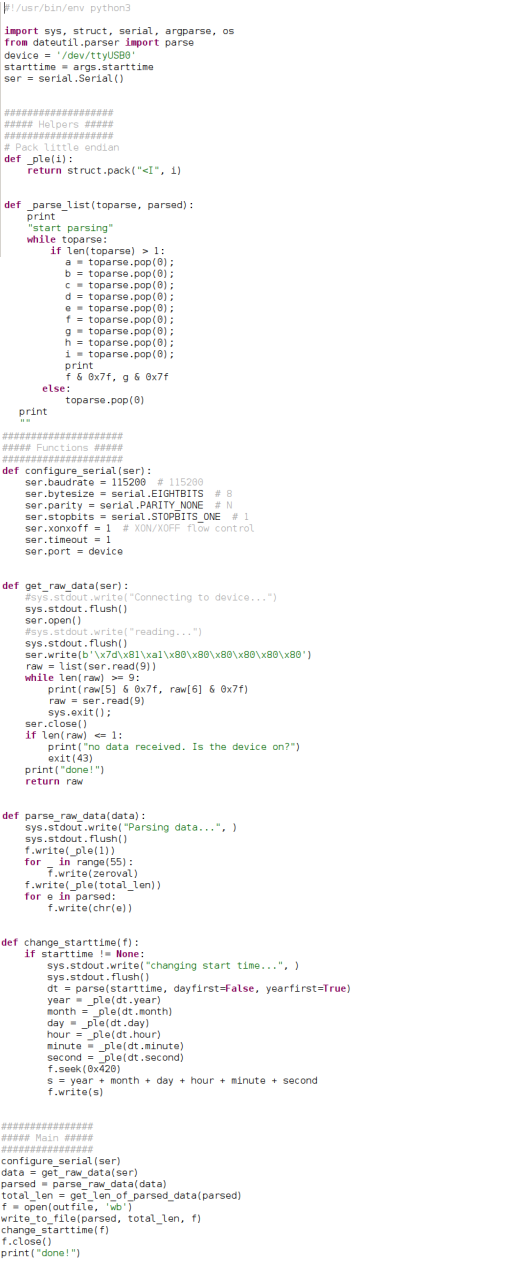
Code for displaying on patient’s web page.



## Code for the Oxygen Level Sensor

## To display on the website.

## Following is python code to get sensor data.



## 

## Code for the heart rate and blood oxygen level

## 

## To display on the website.

## 