ELEC490/498 Final Report: Best Friends

Submitted By

Group 15

Group Members

Ali Irhouma – 20188936 – 19aai1@queensu.ca

Courtney Orcutt – 20155399 – 18cro@queensu.ca

Talia Edwards – 20153662 – 18tle@queensu.ca

Vanusha Vicknesvaran – 20110156 – 17vv19@queensu.ca

Faculty Supervisor

Professor Steven Blostein

Executive Summary

Group 15's approach to creating a smart pet harness known as *Best Friends* came from the lack of devices and applications that monitor the overall health of a pet. The motivation behind the *Best Friends* harness was to build a harness and create an application that can monitoring a pets' health for worried owners and detect any abnormalities in a cost-efficient approach.

An agile approach was used when designing and over the course of this project there were weekly meetings to ensure the team was on track. The project began with research to identify health abnormalities in pets and finding some common health issues that could be used when writing the algorithm. Upon analyzing the problem and the research, a list of goals and criteria was created, and the project was split into subsystems that concentrated on different tasks such as the implementation of hardware, sensor data, API, database, algorithm, and frontend components.

The circuit consists of an Arduino UNO, heart-rate sensor, MPU6050 accelerometer, BMP180 pressure sensor, microphone sensor, and an ESP8266 Wi-Fi Module. Based on the accuracy of readings, the results were sampled to eliminate errors or spikes and produce an output reading every minute for each sensor. The data from each sensor was stored into a concise string that included the heart rate, temperature, step count, and audio samples collected per minute. The string was sent to a database through a PHP API POST request. The API updated the corresponding tables in the database.

Two abnormality algorithms were created. One abnormality algorithm takes in readings from all the sensors identified above. The algorithm will detect any patterns and flag any abnormal readings from the collected data. The abnormalities are based on the research done with the purpose to define normal dog health. The sensitivity will update as a result of user input. Due to the poor readings of the pulse sensor, the team also created a complex algorithm which would uses heart rate frequency ECG signal readings.

The web application begins with a *Welcome* page, where an existing user can login or a new user can register. The *Register* and *Login* pages direct to the user's *Home* page. The Home Page provides information about the product and applications. The *Pet Registration* page lets the user register multiple pets and order the *Best Friends* harness by filling out the form. In the *Monitoring* page, the user chooses which pet they would like to monitor. It includes graphs and tables that provide all the information being collected through the harness as well as abnormality notifications to warn the user of possible health distress.

The cost of the project came to a total of $196.57 which was below the initial prediction of $242.36. The main stakeholders are pet owners and pets. There are numerous safety factors to account for such as weight of harness with hardware components, battery leakage, electricity shocks, and the durability, flexibility, and restrictiveness of design.

All specifications were met except for two which were measuring the respiratory rate and performing abnormality detection on it. The reason for this was because respiration wasn't measured. Test and simulations were validated once each criteria or feature of the harness has met the hardware and software specifications. The result accuracy was compared with the use of data gathered from reliable measurement tools such as an Apple watch.

The project was completed with a harness working as intended. Data being collected from the harness is then transmitted to the database. The data from the database is then accessed by the user from the web application and using the algorithm created, it lets users know if there are any abnormalities or concerns.

Table of Contents

List of Figures

List of Tables

1.0 - Motivation & Background

Dogs are often left at home for long periods of time while owners may be working or running on some errands. At an unexpected time, a dog could be hurt and in distress while the owner may be away and unaware of the event for a long period of time. Over the years, pet owners have attempted numerous alternatives to monitor their pets at home with security cameras as the most common strategy [1]. The camera can capture the pets' movement around the house, but an owner will not be able to know their pet is in distress until after viewing the footage. Another method that has been very popular is microchipping their pet. The only purpose for microchipping is to allow owners to find their pets if they get lost or stolen [2]. There have not been many creations to monitor the well-being of the pet in terms of their heart rate, movement in steps, audio detection, and internal temperature. The limited amount of products that do exist cost between $600 - $900 USD and are only available in the United States. For example, PetPace has released a smart collar worth $600 USD for a three-year subscription to monitor pet vitals such as temperature, respiration, and activity [3]. VetMeasure has realeased a 'MeasureON! Harness' costing $795 USD with an additional monthly $30 subscription fee [4]. On the lower end of the cost spectrum is Invoxia collars which use radar motion sensors to measure movement under the fur and get readings for heart rate, respiratory rate, GPS location as well as activity and sleep monitoring for only medium to large dogs at a cost of $300 a year [5]. In 2021 from the Ontario Veterinary Medical Association, it was estimated that it costs around $3 700 to have a dog per year [6]. The additional cost of paying for a smart harness and upkeeping a subscription would increase the cost of owning a dog by 20% per year. There is a gap in the market for an inexpensive smart pet health monitoring system to put owners at ease.

The motivation behind this project was to build a smart dog harness to monitor pet's health and detect any signs of distress in a cost-efficient way. The objectives were to measure the pet's heart rate, movement, respiration, temperature, and audio to identify and flag any inconsistencies. Common health diseases are diabetes, cancer, epilepsy, heart disease, arthritis, skin disease, and gastroenteritis [7]. Additional health monitoring of vitals for pets with or without these diseases can help pet owners have

better communication and provide data for their veterinarians to better support pets. For example, a pet with hypothyroidism that has elevated body temperature or heart rate consistently may identify to the veterinarian that their thyroid dosage medication may need to be adjusted. If a dog has diabetes, heart rate and respiration patterns can be identified, and insights made after the pet eats or goes for walks. Pets with seizures will have elevated rapid heartbeats. In a 50+ year study released by the Swiss Canine Cancer registry in 2008, of the 122 000 dogs they monitored, more than 50% of the pets got cancer [8]. Pets face almost all of the common diseases that humans face yet are not monitored consistently and as in depth as humans [7]. Pets face numerous health battles and through monitoring and identifying abnormalities early, the treatment and health of a pet could be improved drastically.

2.0 - Design

The design was developed using an agile approach to increase the speed and productivity of the team. Weekly team scrum meetings began sprints that would implement multiple features from the backlog that are described in Table 1 and Table 2. The design began with research into the health abnormalities that impact dogs depending on their size and types. Normal heart rates and temperatures for large and small dogs were also noted to be used in the abnormality algorithm. Some common heart issues were also explored to allow the algorithm to identify possible patterns. A summary of the research can be seen in Appendix A - Health Abnormality Parameters. The design was also based on the idea generation mind map found in Appendix B- Progress Planning and Idea Generating. The design was separated into subsystems focusing on hardware, website frontend, data processing, the database, and the algorithm. Each subsystem was tested individually through the use of unit tests that would fulfill the preset requirements. When the subsystems were merged, integration testing ensured the project worked as intended. The subsystems and their relationships are described by Figure 1.

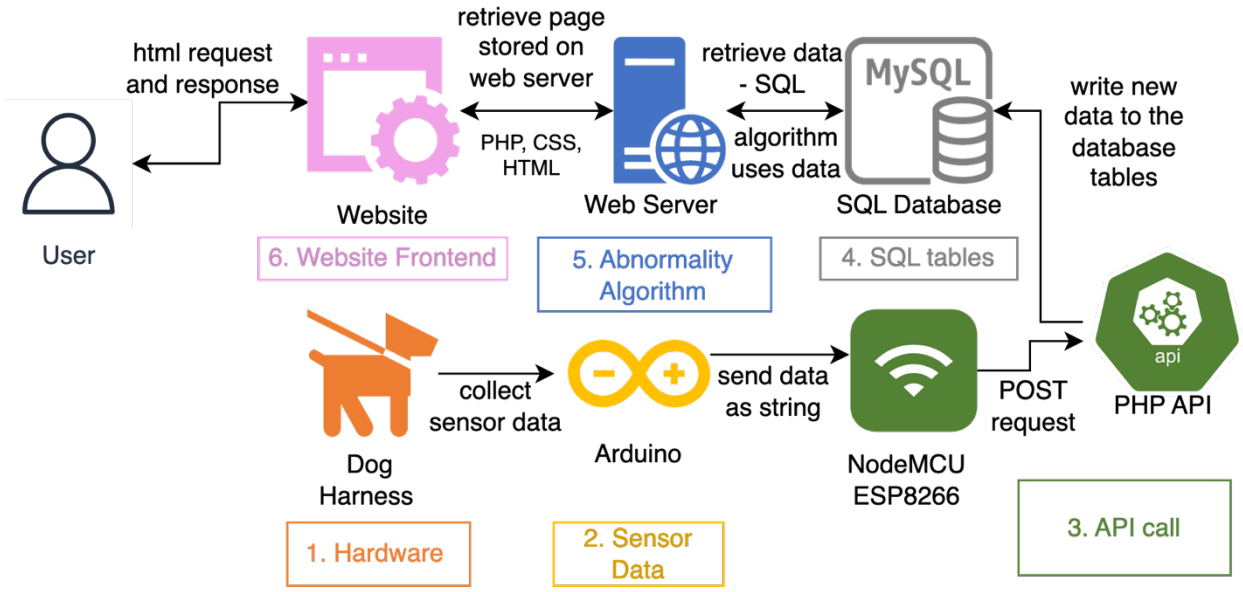

Figure 1: The subsystems of the health monitoring device

After the problem was thoroughly analyzed the specification tables below were created to fulfill the goal of creating a dog harness to monitor the health of dogs. Notably, the tables were modified with the purpose of allowing readers to identify which subsystem each specification impacted with the added column called subsystem. The key decision to remove the location detection feature was made before implementing the specification table. After working on the design for a few weeks, the team also decided to focus on the algorithm for the heart rate since a lot of data had been gathered to assess the health of a dog's heart. The projects main constraint was the performance of sensors since they played a large role in determining the accuracy of the abnormality algorithms.

Table 1: Hardware System Specifications

Specifications		Target value	Tolerance	Subsystems
1	Overall harness weight including the sensors, fabric, and Arduino	*500 g*	*+ 500 g*	Hardware
2	Reliable WIFI connection	Constant connection	Loss of connection below 1 min	API
3	Data processing rates with minimal delays including the time it takes for the sensor data to be saved in a database by the Arduino	Maximum 5 min delays	+ 1 min	
4	Low Power Consumption resulting in a long battery life	Battery lifespan of 7+ hours	- 3 hours	Hardware

5	The heartbeat monitoring sensor must detect all heartbeats per minute in the range from 50 to 130 heartbeats/min.	All heartbeats/min	+/- 10 heartbeats per min	Sensor Data
6	The respiratory sensor must detect respiration per minute in the range of 5 to 35 breaths/min.	All breaths/min	+/- 5 breaths per min	
7	The temperature sensor must accurately measure body temperature in the range from 99°F to 104°F.	Current temperature with decimal values.	+/- 1°F	

Table 2: Software System Specifications

1	**Functional requirements**	**Subsystems**
1.1	Application to give the user access to data and settings	Website Frontend
1.2	Algorithm to detect common heart rate anomalies (e.g., arrythmia) in real-time	Abnormality Algorithm
1.3	Algorithm to detect common respiratory rate anomalies (e.g., obesity) in real-time	
1.4	Algorithm to discover sudden spikes in data in real-time	
1.5	Algorithms adapts to user feedback (Users are notified of abnormalities and can approve or disapprove of a detection)	
1.6	Application considers users preference such as abnormality sensitivity levels or type of abnormalities detected	Website Frontend
2	**Interface requirements**	**Subsystems**
2.1	Data page to display the temperature, respiratory, movement, and heart rate data in the form of graphs	Website Frontend
2.2	Notification page to display notifications of abnormality detections	
2.3	Settings page to adjust dog type, age, weight, size, name, and user preferences	
3	**Performance requirements**	**Subsystems**
3.1	Retrieval of data from a database is under 2 minutes	Database
3.2	Change of the application page is under 10 seconds	Website Frontend, Database
3.3	Real-time notifications are based on algorithm results	Abnormality Algorithm, API, Database
3.4	After a training period, the chosen abnormality algorithm should aim to reduce the rate of false detection under 30%.	Abnormality Algorithm
3.5	The number of missed detections (the abnormality is never detected) should be extremely low. The user should be able to set the sensitivity level of the device to reduce or increase the number of notifications received.	Abnormality Algorithm, Website Frontend
3.6	Performance will be measured by comparing the link between false detection and missed detections. The algorithms that reduce the number of missed detections while keeping the number of false detections low will be chosen to maximize the probability of correct detection.	Abnormality Algorithm
3.7	Algorithms will use parameters such as healthy heart rate, respiration frequency, and body temperature to find abnormalities. Algorithms will use unsupervised learning	Abnormality Algorithm

	to detect any extremes in data and will use models of common health issues.	

The backend subsystems were designed in depth before beginning implementation. The hardware system sketch in Figure 2 was created to indicate the ways to connect each sensor to the Arduino. Each sensor was tested individually before the subsystem was fully assembled to detect any mistakes.

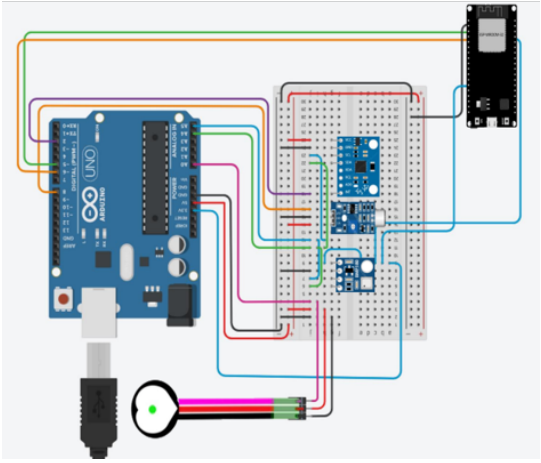

Figure 2: The sensors, Arduino, and NodeMCU ESP8266 circuit

The input collected from the previous circuit was sent to the database using the API subsystem which was designed using openapi3 documentation and the swagger editor as illustrated in Figure 3. As can be observed, the figure highlights the request body for the post request which includes the heart, movement, temperature, audio, and date for the entry into the database. A 200-response code means the request was successfully written into the database. In the case where the request would fail to be written or the host could not be found, a 500 internal error would occur.

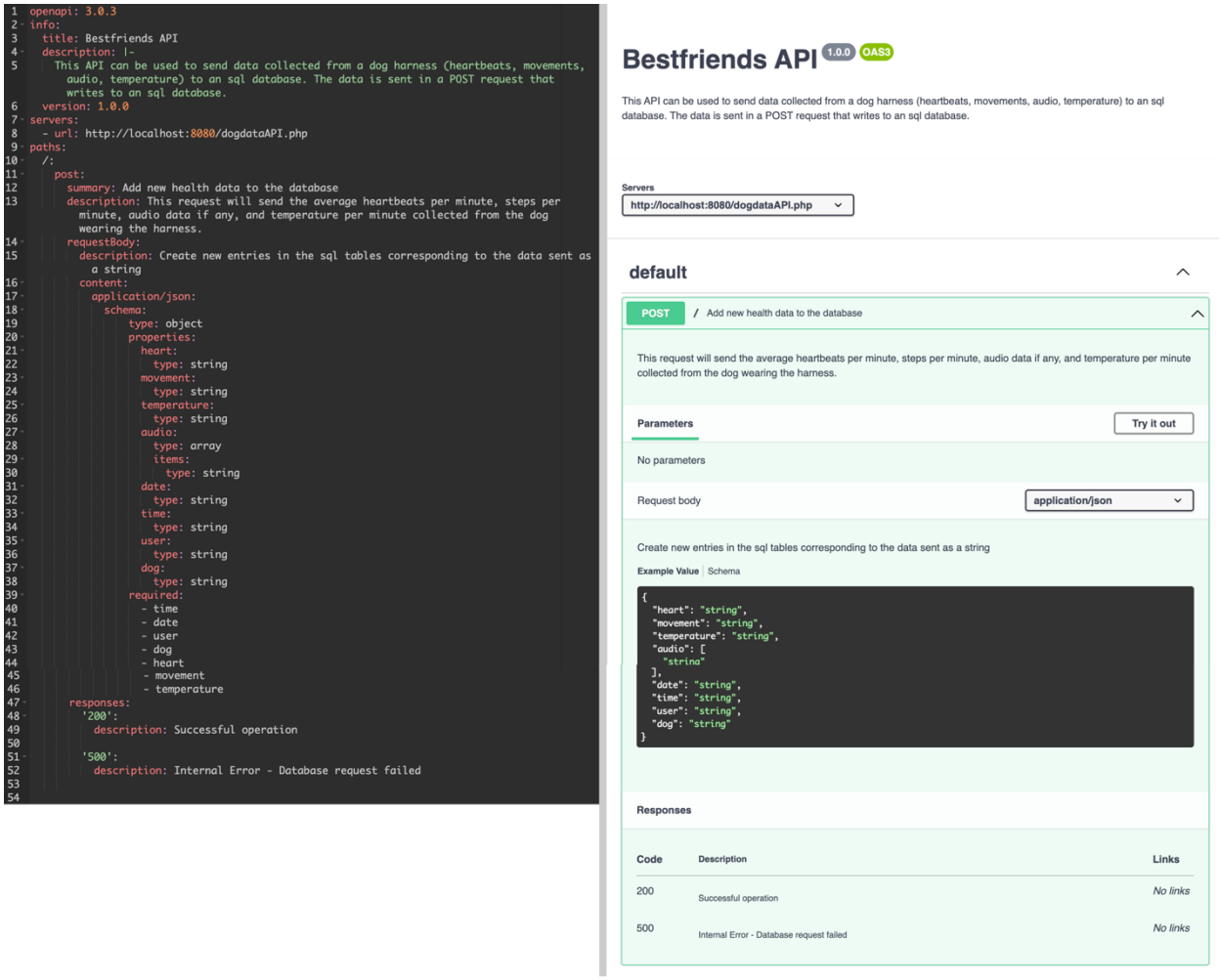

Figure 3: The API documentation

Furthermore, the database had to be designed meticulously to prevent the need to modify it during implementation which would take more time and could reduce the efficiency of the final product. The different tables and their attributes were organized in the entity relationship diagram shown in Figure 4.

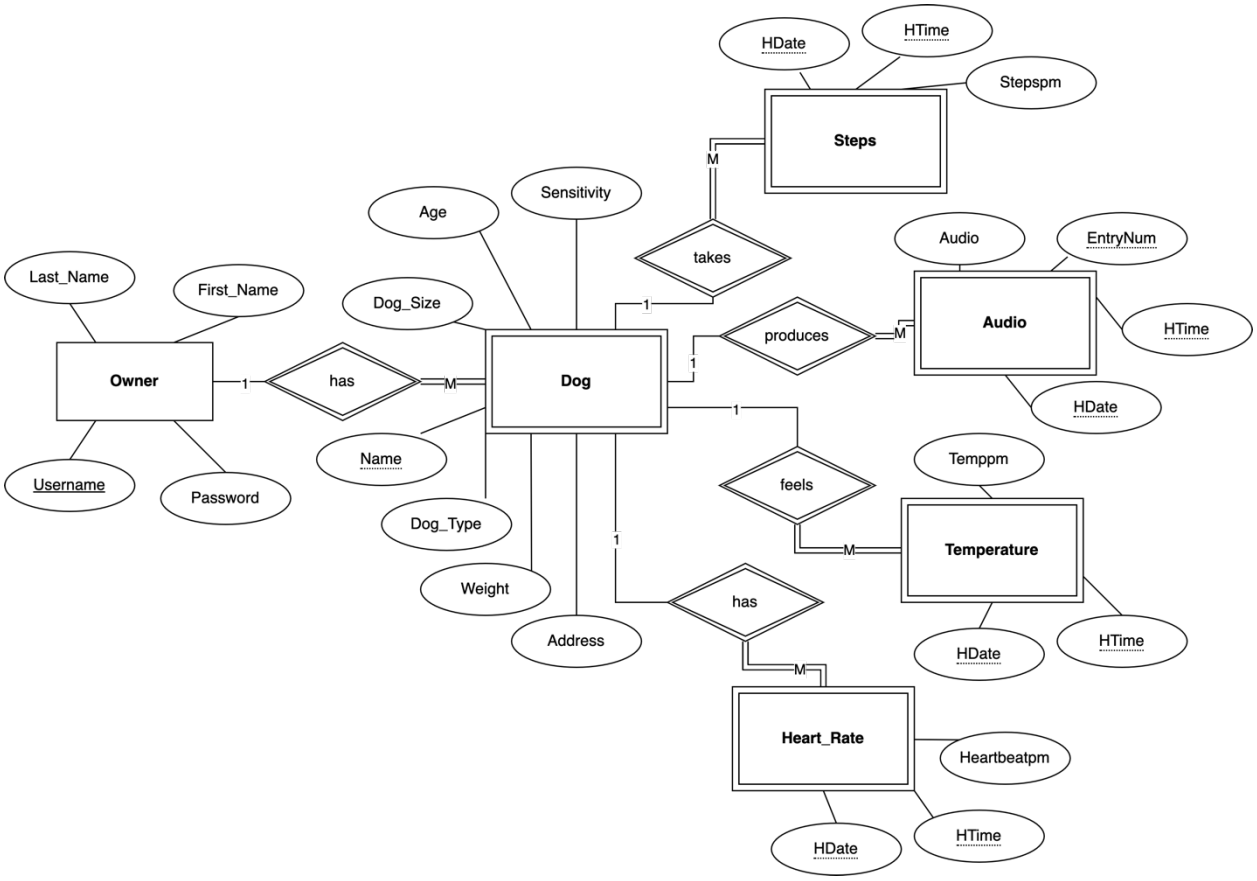

Figure 4: Database ER Diagram for Bestfriends

The frontend subsystems such as the algorithm and the website frontend designs relied on the requirements. For instance, the website frontend had to include a page to gather settings, a page to display data in the form of graphs and tables as well as notifications about abnormalities. Moreover, the abnormality algorithm had to receive input from sensors (temperature per minute, sound per minute, heart rate per minute) and output any detected abnormalities at specified times.

The design process was recorded through 5 main sprints to implement the functionality of the application and the backend. The progress of the team is summarized in Table 3.

Table 3: The design process and progression

Sprints of Implementation	Update
1	• Find the required sensors • Purchase materials

	• Research to find data of existing dog heartbeats and health issues • Research Wi-Fi connectivity • The design planning stage is completed
2	• The material arrived • The hardware subsystem is assembled: each sensor is added to the circuit • Data is still raw and not in a string to be sent in a request • The NodeMCU ESP8266 can connect to WiFi • A php API is starting to take shape (it accepts post requests with a simple body) • The database text file is created to match the ER diagram • A simple algorithm is created (accepts heartbeats per minute)
3	• Frontend begins and uses the database information for login • Minor setbacks making the web server public • Resoldering of the heartbeat sensor is needed since the connections are loose • The API is completed and tested using the Postman application (post request with complete request body can be sent and written to the database)
4	• Frontend is completed (notifications are added and diagrams display the backend data from the database) • A complex algorithm is created however it requires a better heart monitoring sensor (it is tested using data from a veterinary database) • Hardware is completed and data is parsed into a string • The string is sent to the NodeMCU ESP8266 which can make a POST request • The database gathers the data through the POST request and responds to sql statements from the website frontend
5	• Soldering the circuit • Testing: abnormality algorithm, data collection, API calls, frontend webpages • Result analysis to determine the accuracy of the detections

3.0 - Implementation

The implementation of the project was divided into four different subcategories which included the hardware, API and database implementation, frontend website and the algorithm analysis.

3.1 - Hardware Analysis

The sensors and components used for the harness had to be compatible with the Arduino UNO. The complete Arduino hardware code is provided in the following link that provides how the heart rate,

accelerometer, pressure, Wi-Fi module, and microphone sensor are used: [ELEC498-BestFriends/Hardware_Code.ino at main · TaliaEdwards11/ELEC498-BestFriends [9]]

Pulse Sensor

The pulse sensor (SEN-11574) can detect and read heart rate pulses in BPM through skin contact. Photoplethysmogram (PPG) is the process that occurs when the sensor transmits a green LED light that is absorbed by the volume of blood within the tissue and reflected to the photosensor by the skin interaction. Depending on how red the blood is within the tissue, there would be a higher absorption of the LED light, thus the change of reflected light to the sensor determines the blood flow rate in BPM [10].

The sensor consisted of three leads: ground (GND) which is connected to the ground pin, the voltage source of 5V (VCC), and the signal output (S) that is connected to the analog input (A0) to produce readings from the photosensor [10] [10].

```
//HeartBeat Sensor

if(millis() - beat_captured_now > 20){              // Sampled heartbeat duration

  if (pulseSensor.sawStartOfBeat()) {               // Checking constantly is a heartbeat occured
    int myBPM = pulseSensor.getBeatsPerMinute();    // Storing the reading as an integer in myBPM
    total_heart_beats = total_heart_beats + myBPM;
    //Incremeting number of times BPM recorded
     how_many_times_did_heart_beats_get_recorded = how_many_times_did_heart_beats_get_recorded + 1;
  }
  beat_captured_now = millis();

}
```

Figure 5: Heartbeat Sensor code within the void loop function

After the pulse sensor has been connected to the appropriate pins, a portion of the code in Figure 5 showcases that if there is a detection of a heartbeat, the results received from the photosensor will be read and displayed. However, the reading received was initially extremely inaccurate. Based on observation, it was preferred to resolder the leads to the sensor as it was done poorly by the manufacture and sample the results from the pulse sensor, which potentially eliminated any spikes or outlier readings. Figure 5 above demonstrates that the results taken are sampled over an appropriate interval.

MPU6050 Accelerometer/Gyroscope Sensor

The MPU6050 accelerometer is able to measure the acceleration and angular velocity in all three axis, roll (x-axis), pitch (y-axis), and yaw (z-axis) when in motion. Based on the change in acceleration, the number of steps taken can be calculated [11].

The accelerometer is configured to the following leads connected to the pins: 5V voltage source (VCC), ground pin (GND), a serial clock pin (SCL) for I2C module that sends a pulse at a specific interval from the Arduino board and connected to the A5 pin, and a serial data pin (SCA) for I2C module that acts as the data line and connected to the A4 pin [11] [12].

```
  //Checking for acceleration change in the z-axis. If there is change in the acceleration, then the step count increments
    if( accelerometer_z < 0 )
        flag = true;
    else if( flag == true && accelerometer_z> 0 )
    {
        flag = false;
        total_steps_taken = total_steps_taken+1;
    }
}
```

Figure 6: Code for the number of steps counted in the void loop function.

After incorporating the accelerometer to the circuit, the code in Figure 6 demonstrates how the number of steps is computed. When the sensor is in motion and there is a change in acceleration in a certain axis, the accelerometer is able to detect that as one step taken. This has been successfully simulated by moving the sensor back and forth, which accurately counted the number of steps taken. As shown in the code above, when there is a change in acceleration in the z-axis, the number of steps taken would increment.

BMP180 Barometric Pressure Sensor

The pressure sensor was an additional feature that was intended to measure the respiratory rate. The sensor can record the environmental pressure, temperature, and the altitude of the surrounding environment. The BMP180 has the capability to read temperatures ranging from -40^{O}C to 85^{O}C, pressure ranging from 300hPa to 1100hPa, and altitude ranging from 0 to 30,000ft, providing a large capacity to tolerate extreme environmental surroundings and the pet's body temperature [13].

The sensor consists of four pins: A 5 V input (VCC), ground pin (GND), an I2C serial clock pin (SCL), and an I2C serial data pin (SDA), where SCL and SDA are connected to analog pins A5 and A4 respectively [13].

```
if(millis() - reading_captured_now > 500){
    float temp = bmp.readTemperature();
    float pressure = bmp.readPressure();
    float altitude = bmp.readAltitude();
    reading_captured_now = millis();

    total_temp_taken = total_temp_taken + temp;
    how_many_times_did_temp_get_recorded = how_many_times_did_temp_get_recorded + 1;

    total_pressure_taken = total_pressure_taken + pressure;
    how_many_times_did_pressure_get_recorded = how_many_times_did_pressure_get_recorded + 1;

    total_altitude_taken = total_altitude_taken + altitude;
    how_many_times_did_altitude_get_recorded = how_many_times_did_altitude_get_recorded + 1;
}
```

Figure 7: Pressure, temperature, and altitude readings in the void loop function

Figure 7 demonstrates how the results for the temperature (temp), pressure, and altitude are taken. During simulation, the results from the sensor were very accurate, even without the additional data sampling.

Sound/Microphone Sensor

The microphone sensor was another essential component that aided the performance of the harness by including a feature that is able to record the pet's sounds such as barking, which is used to indicate any abnormalities in barking behavior. The microphone is also able to detect sounds from the pet's environment as well.

The sensor consists of three pins: a 3.3V voltage source (VCC), a ground pin (GND), and an output pin that is connected to digital pin 8. A digital pin is used to produce output status of either HIGH or LOW when sound is not detected or detected respectively [14]. In addition, the sound sensor has a built-in potentiometer, which allows to adjust the sensitivity of the microphone manually [14].

```
// Sound recorded if above 0, the code will be sampled and placed into the array
 if (millisElapsed >= SAMPLE_TIME && index_for_sampleBufferValue_array < 20) {
  if(sampleBufferValue>0){
    sampleBufferValue_array[index_for_sampleBufferValue_array] = sampleBufferValue;
    index_for_sampleBufferValue_array = index_for_sampleBufferValue_array + 1;
  }
  sampleBufferValue = 0;
  millisLast = millisCurrent;
  }
```

Figure 8: Code for the microphone readings that stores in an array in the void loop function

After integrating the sensor to the Arduino and uploading the code, the microphone is able to detect incoming sound over a period of time, which is then printed in an array form. However, the database is not capable of providing a large array; therefore, eliminating unnecessary recordings was an effective strategy. As presented in the portion of code in Figure 8, if sound was detected, the result will be considered; otherwise, the result is disregarded. To further limit the capacity of the array in the database, the detected sounds are sampled, then stored in the array called sampleBufferValue_array.

NodeMCU ESP8266 Wi-Fi Module

The Wi-Fi module was used to send the data gathered by the Arduino to the MySQL database. In fact, the Arduino sent a string containing the average heart rate, temperature, step count, and noise samples per minute to the serial input of the NodeMCU ESP8266. The NodeMCU ESP8266 was responsible for establishing a Wi-Fi connection as well as sending a POST request to the database API. The API would send SQL requests to insert data into each corresponding table.

The Arduino was programmed to send data to Serial through two specified pins with the command: SoftwareSerial espSerial(5, 6). Afterwards, the string would be sent with the Serial.print(string) command. The NodeMCU was separately coded. It would parse the string input using a loop that checked if Serial.available() was true. All strings began with the '*' flag and would end with ';'. The code connected to Wi-Fi, then parsed the string, and finally sent a http.POST(temp) request where temp was the request body. If the request is successful, it returns a 200 response code. The full code is available in the file named wifi_NodeMCU_ESP8266.ino:

https://github.com/TaliaEdwards11/ELEC498-BestFriends/blob/main/wifi_NodeMCU_ESP8266.ino

By including all components in an integrated circuit as shown in Figure 10, the results from all the sensors are stored in one string, including the microphone array, and are all sent to the database via the Wi-Fi module. Figure 9 showcases how the overall results are sent to the database.


```
{"user": "Billy123",
"dog" : "Doguino",
"date" : "2023-06-24",
"time" : "03:46:51",
"heart":112 ,
"steps":0 ,
"temperature":23.71,
"Sound":[22,61,51]}
```


Figure 9: The string sent by the Wi-Fi module to the sql API

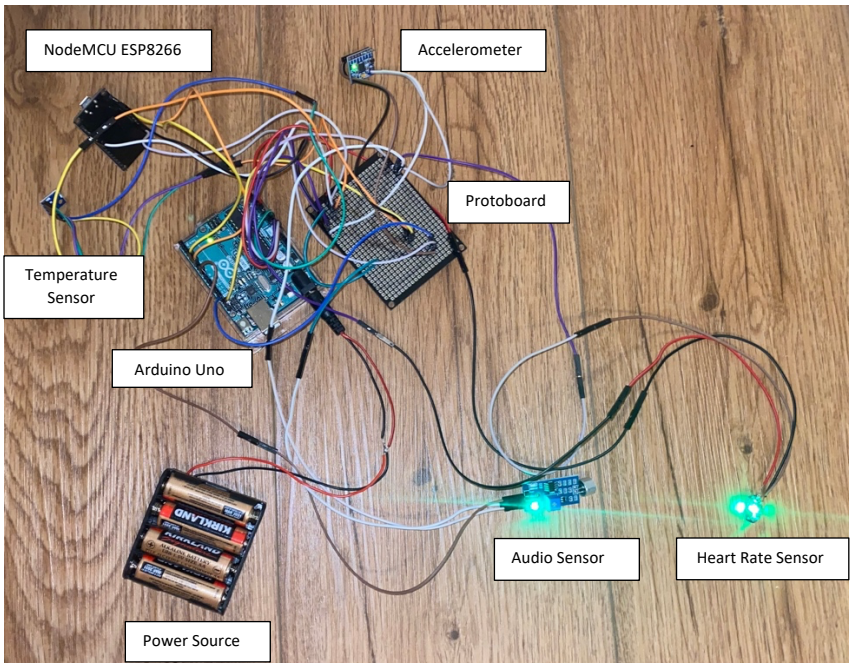

Figure 10: The prototype circuit with labels

3.2 - API and Database Implementation

The data sent in the POST request is received by a PHP API file called dogdataAPI.php. The code can be accessed with the following link:

https://github.com/TaliaEdwards11/ELEC498-BestFriends/blob/main/dogdataAPI.php.

The code checks the POST request input received from the NodeMCU ESP8266 in JSON format through the request body. It verifies that the data is valid and parses the body then proceeds to submit the correct SQL insert statements to update the database tables. The data processing rate is 1 minute since the API file immediately inserts the dog health data into the database. As soon as the request is received, the database updates in real-time.

The software application named XAMPP hosted the MySQL database that was used to store the tables shown in Figure 4. The SQL database was created by importing the following text file containing the SQL code required to create tables with their specific attributes as well as insert sample values:

https://github.com/TaliaEdwards11/ELEC498-BestFriends/blob/main/bestfriends.sql

The results appear in the database tables in the format pictured in Figure 11. These tables are not accessed directly by the user. The data is illustrated with the use of plots as it will be demonstrated in the frontend section of the implementation.

HDate	HTime	Heartbeatpm	Dog_Name	Owner_UserName
2023-03-23	15:03:25	127.00	Doguino	Billy123
2023-03-23	15:04:26	141.00	Doguino	Billy123
2023-03-23	15:05:26	185.00	Doguino	Billy123
2023-03-23	15:06:26	165.00	Doguino	Billy123
2023-03-23	15:07:30	157.00	Doguino	Billy123

Figure 11: Sample data from the heart rate table from the database

3.3 - Frontend – Website

All the frontend website code was created and uploaded through the application XAMPP. The frontend code for the website can be found at the following link:

https://github.com/TaliaEdwards11/ELEC498-BestFriends

The overall concept of the front end was to give the owner of the pet a place where they could login and check in on their dog in all aspects being detected by the Best Friends harness. As mentioned earlier, this includes heart rate monitoring, counting the number of steps, the internal temperature of the pet and finally listening to the audio of the pet to detect any unusual sounds.

The website begins with the *Welcome* page, where the owner can find more information about *Best Friends*, the harness and the services provided. This page has three buttons that lead to the *Registration* page, *Login* page and the *About Us* page, as seen in Figure 12.

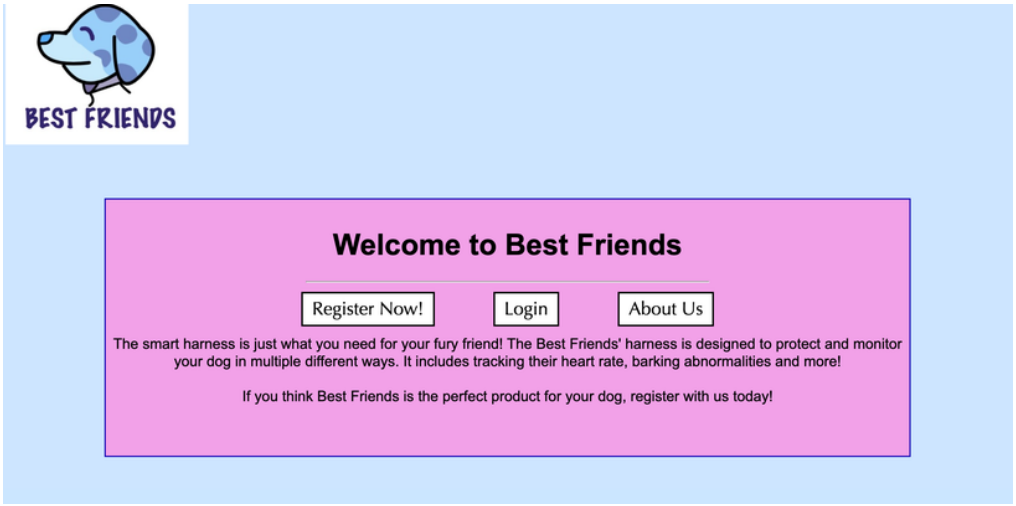

Figure 12: Welcome page on the web application

The *Registration* and *Login* page follow the same styling format. The *Registration* page asks the owner for their first name, last name, a username, and a password. The PHP code, shown in Figure 14 ensures that the username is not already taken and the password follows all the requirements which includes a lower-case letter, an upper case letter and the specific length. The *Login* page asks the owner to enter their username and password, as seen in Figure 13.

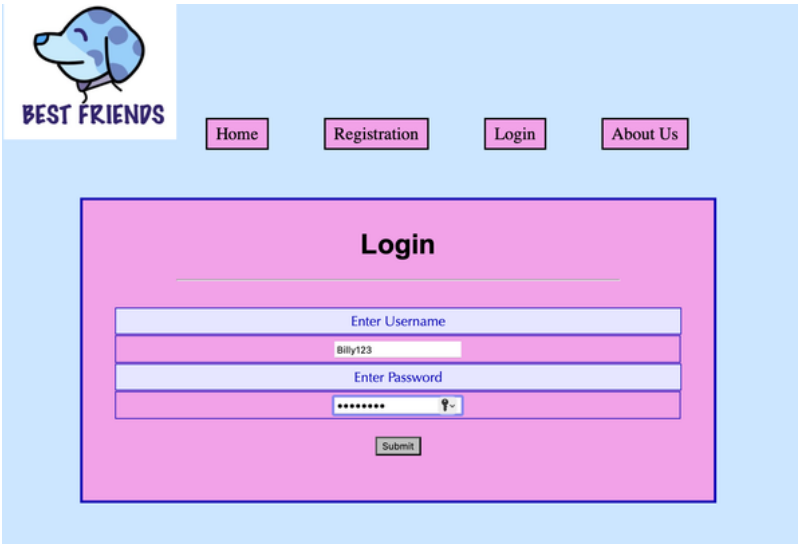

Figure 13: Login page as seen on the web application

Finally, the *About Us* pages gives the owner a little insight on the engineering team that created *Best Friends*.

```php
<?php
    if (isset($_POST['registration'])){
        $FName = $_POST['FName'];
        $LName = $_POST['LName'];
        $username = $_POST['username'];
        $password = $_POST['password'];

        //check that this owner does not already exist
        if(ctype_alnum($username) && strlen($username) > 0 && strlen($username) <=100 && $FName!="" &&  $LName!="" && $password
            !="" &&   ctype_alpha($FName) && ctype_alpha($LName)) {
```

Figure 14: Code written in PHP to ensure the a owner can successfully register

Once logged in, the owner is taken to the *Home* page as shown in Figure 15 where there are two buttons where the owner can register their pets and for monitoring.

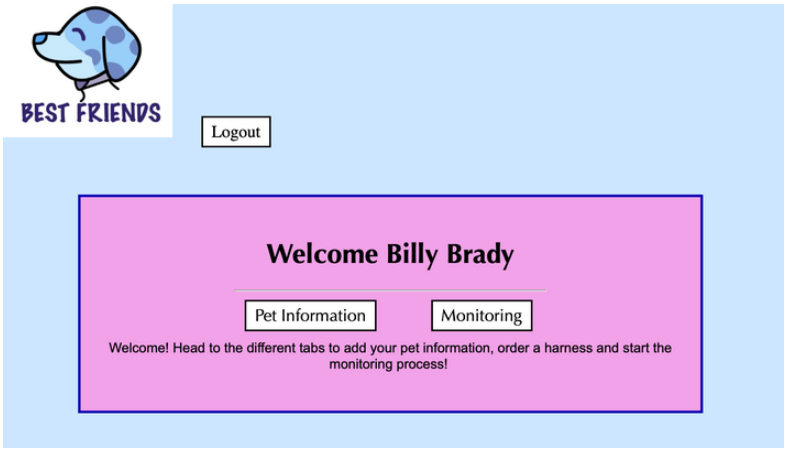

Figure 15: Home page of the web application

The owner can register multiple pets at a time. The *Dog Registration* page asks the owner for the name of their pet, the dog type, more specifically if their dog is one of the ones listed in the drop-down menu which include Bulldogs, Pug, Pekinese, Saint Bernard, Great Dane, Mastiffs, Boxers, Doberman Pinschers. From research from our Blueprint report, it was determined that these are the most likely breeds with a chance of a medical condition. The page also asks the owner for their pets age, weight and sensitivity to food or medications as well as their address to send the owner their harness, as shown in Figure 16.

Figure 16: Dog Registration page as shown on the web application

All this data is relevant as it helps improve the overall algorithm when tracking the various parameters attached to the harness. Similarly, to the *Registration* page, the *Dog Registration* page ensures the owner does not register the same pet twice, by using similar logic as seen in Figure 14.

One of the main purposes of the website is so the owner can track their pets' well-being. This has been accomplished by providing the owner with a series of tables and graphs on a tracking page. Once entered on this page, the owner will choose from a drop-down menu the pet they would like to see the data for. Once the submit button is clicked, a series of graphs and charts will appear for each parameter being tracked, as seen in Figure 17 and Figure 18.

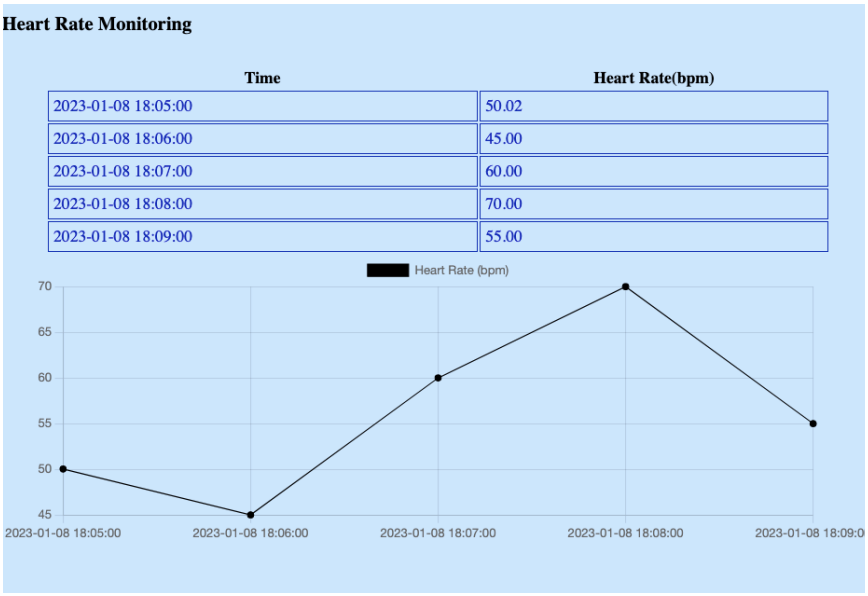

Figure 17: Graphs and tables for heart rate monitoring

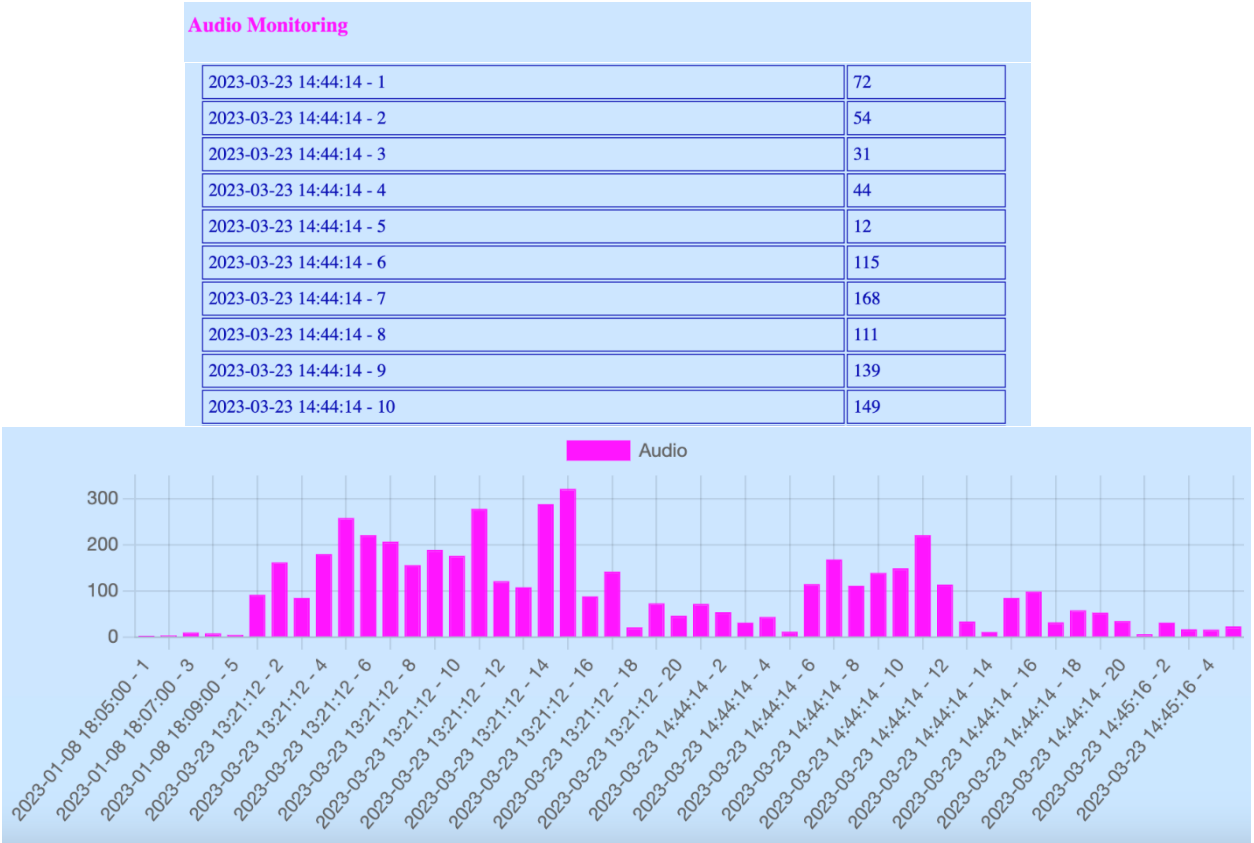

Figure 18: Audio monitoring data collected in a table and bar graph

The graphs and tables were created using the following code shown in Figure 19. Four different scripts were created for the four different graphs shown to the owner. This snip of code takes the time and

the heart rates from the harness and prints it directly into the graphs and tables in real time. This step is repeated for the remaining three items monitored by the harness.

```
<script>
    var ctx = document.getElementById('myChart').getContext('2d');
    var myChart = new Chart(ctx, {
        type: 'line',
        data: {
            labels: <?php echo json_encode($time, JSON_NUMERIC_CHECK); ?>,
            datasets: [{
                label: 'Heart Rate (bpm)',
                data: <?php echo json_encode($heartRate, JSON_NUMERIC_CHECK); ?>,
                backgroundColor: 'rgba(0, 0, 0, 1)',
                borderColor: 'rgba(0, 0, 0, 1)',
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true
                    }
                }]
            }
        }
    });
</script>
```

Figure 19: Code used to create the graphs and tables shows on the website

Finally, the *Monitoring* page will also include notifications to let the owner know if there have been any abnormalities detected based off the data collected from their pets' harness and the algorithm created.

3.4 - Algorithm Analysis

The components built into the dog harness measure the heart rate in BPM, temperature, pressure, microphone noise, and step count. The team created an algorithm to detect abnormalities in these health measurements and flag if the dog is in motion using the accelerometer sensor. The team also created a more complex algorithm which takes heart rate ECG frequencies from a reliable health database from PhysioNet [15] [16]. PhysioNet is a medical research data page managed by the Massachusetts Institute of Technology. Since one of the key challenges of this project was not having reliable data readings from the pulse sensor, the complex algorithm would be paired with a high-end expensive ECG frequency signal sensor before the smart pet harness would go to market. This would ensure that the readings are reliable and accurate.

The regular heartbeats algorithm takes in the input parameters of the heartbeats array, the timestamps of the corresponding heartbeats array, the size (n) of the heartbeats array, the dog weight, dog breed, dog age, step count, and the sensitivity value. The algorithm will output an array of the outliers (high maximum heart rates or low minimum heart rates) and the timestamp that the abnormality occurred. Depending on the dog breed and size, different heartbeats ranges are acceptable. For example, small dogs and puppies have a normal heartbeat per minute range between 120 - 160 bpm, whereas large dogs have a normal heartbeats range between 60 - 120 bpm, similar to human beings [17]. As well, the size and breed of the dog is helpful for the algorithm to give the most accurate heart rate pattern flags such as sinus tachycardia, sinus arrhythmia, atrial fibrillation, atrioventricular block, and premature beats. For sinus tachycardia, where the heartbeat beats at a faster rate, puppies usually will have heart rates above 220 bpm, whereas giant breeds would have heart rates flagged above 140 bpm. Another reason for the importance of the breed as an input parameter to the algorithm is that certain breeds have preexisting conditions or are predisposition to have certain health concerns. For example, Pugs, Bulldogs, Boxers, Doberman Pinschers, and Great Danes are all breeds prone to have irregular abnormal heartbeats [18]. Breeds such as bulldogs are predisposition to obesity and trouble breathing.

The regular algorithm will first read in all the inputs and depending on what the owner has specified as their pet's breed and weight, the algorithm will personalize to the dog. For example, if an owner has a pug that weighs 20 pounds, then the normal allowed heart rate range is adjusted to them. The minimum expected heart rate value would then be 120, and the maximum would be 160. First the algorithm will sort the array of heartbeats and order them all from minimum bpm to maximum bpm. The maximum and minimum values are then stored in the algorithm as boundary limits. As well, the algorithm will check for a weight concern flag if the bulldog is deemed overweight as seen in Figure 28. If weight is a concern, being either too high or too low then it will factor into the sensitivity of the algorithm. Below in Figure 20 is the beginning portion of the algorithm tuning to the pet's input

characteristics and searching for the abnormalities and pushing them to a string to show on the interface frontend.

```
# algorithm will update based on users input of abnormality flags being true or false which affects the sensisitivity
function getAbnormalities($MyArray, $n, $MyTimeArray, $sensitivity, $dogWeight, $dogBreed, $age, $sizeDog, $stepCount){

    if ($dogBreed == "Bulldog" || $dogBreed == "Chihuahuas" || $dogBreed == "Maltese" || $dogBreed == "Dachsund" || $dogBreed == "Pomeranian" || $
        dogBreed == "Pug" || $dogBreed == "Boston Terrier" || $dogBreed == "Shih Tzu" || $dogBreed == "Yorkshire Terrier" || $sizeDog == "Small" ){

        // depending on breed set minimum and maximum bpm value
        $minValue = 120;
        $maxValue = 160;
        $count = 0;

        // check the weight
        checkWeightConcerns($stepCount, $dogBreed, $dogWeight);


        // pre-existing conditions check
        checkExistingHealthConcerns($dogBreed, $dogWeight, $age);

        // check if greater than 160 bpm and depending on sensitivity range
        // if higher than should be. For example, the higher the sensitivity
        // rating, bpm of 155 are ok but not ok with a sensitivity of 1
        $results = array();

        foreach($MyArray as $key=>$e){
            $i = 0;
            if($stepCount[$key] < 15 ){

            $difference = $e - $maxValue;

            //echo $difference;
            //echo $sensitivity;

            $threshold = 30;

            if($e>$maxValue && $difference > $threshold/$sensitivity){


                    $str = "Abnormality detected, specifically the high bmp rate of : ".$e. " at time reading of : ".$MyTimeArray[$key]."";
                    array_push($results, $str);


            }

            }
        }
```

Figure 20: Beginning of getAbnormalities() function that checks if the inputted dog breed and weight is classified as a small dog and sets the threshold limits, checks for abnormalities and pushes them to the frontend to have sensitivity adjusted.

Then the algorithm will check the whole dataset of heartbeats to flag abnormality patterns such as rapid high heartbeats. For that same pug since, it is considered a small dog, the algorithm will check if any of the heartbeats are over 160, the set maximum limit value (maxValue). Then depending on the sensitivity specified by the user and the amount of false positives checked off in the interface, the algorithm will adjust. For example, if the user inputted that the algorithm is detecting some false positives, the sensitivity will automatically be adjusted so that it goes higher on a scale of 1 - 3. Figure 23 and Figure 25 the interface of what the user will see when the sensitivity in the backend adjusts. If the dog naturally has heart rates that are 165 or 163 and is healthy and normal, then the sensitivity adjustment will make the threshold limits for acceptable heartbeats larger so that they won't be flagged. On the other hand, if the user thinks the algorithm isn't sensitive enough, it can have the sensitivity adjusted to 1 where bpm that are 150 or 155 would start to be flagged. If the algorithm notices higher heartbeats that seem abnormal, it flags a check to see if the step count is high for small and large dogs and has the threshold for acceptable heartbeats adjusted since the dog is in motion. Below are what the user will see from the

abnormality algorithm on the frontend from Figure 21. The pet owner has the ability to identify what abnormalities should be ignored and the sensitivity will update as a result.

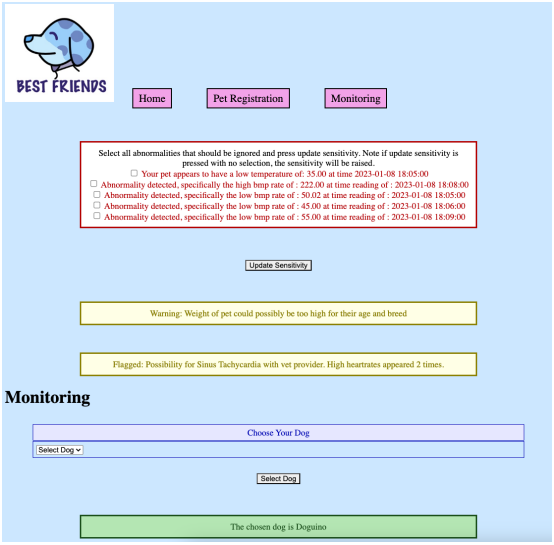

Figure 21: Retrieve abnormalities on the monitoring page.

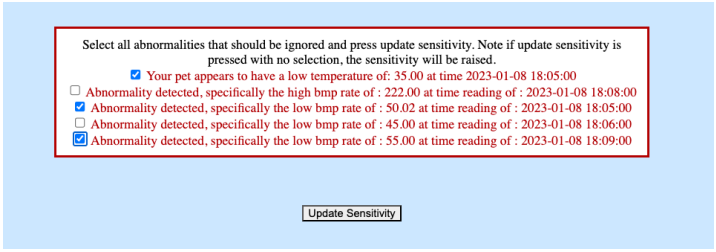

Figure 22: Select the false detections of the abnormalities on the monitoring page.

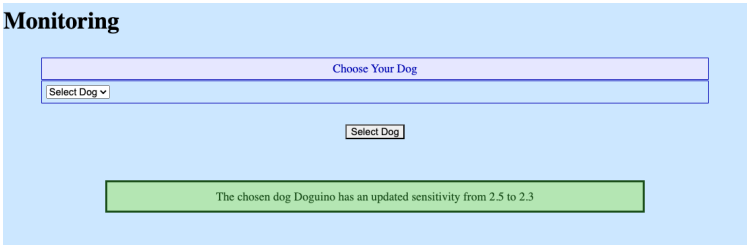

Figure 23: Update the sensitivity based on the user's input of the abnormalities to be ignored.

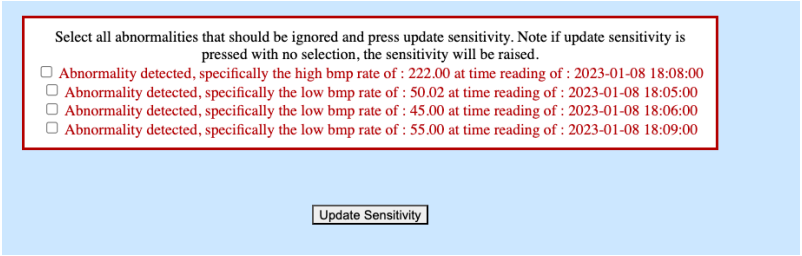

Figure 24: Retrieval of the abnormalities based on the updates sensitivity value.

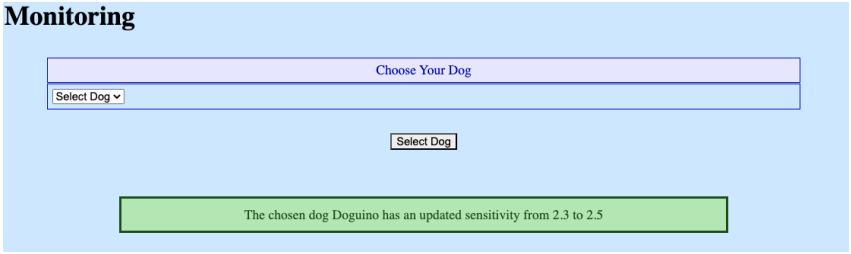

Figure 25: Update sensitivity with no selection of false detections (abnormalities).

From the size and breed of the dog, the algorithm isn't affected by differing inputs for the respiration rate, panting rate, and body temperature. For example, respiration is usually between 10 - 30 breaths per minute and normal body temperature for all dogs is between 38.3 °C to 39.2 °C [19]. Preexisting conditions will change these readings to be higher if a dog is experiencing hypothyroidism or diabetes. The algorithm will check to ensure that the internal body temperature isn't too high or too low for the dog, which could be a concern on especially hot days as seen in Figure 26. The algorithm works in tandem with user input and sensitivity adjustments as shown in the front-end analysis section of output graphs. This section has highlighted how functional requirements 1.1, 1.2, 1.4, 1.5, and 1.6 have all been integrated, except for 1.3 which runs a respiratory algorithm.

https://github.com/TaliaEdwards11/ELEC498-BestFriends/blob/main/algorithmTesting.php

```php
function temperatureCheck($MyTimeArray, $Temp){
        // maximum temperature is 39.2
        // minimum temperature is 38.3

    $tempMinLimit = 38.3;
    $tempMaxLimit = 39.2;
    $results = array();
        $count = 0;
        foreach($Temp as $e){
            if(floatval($e)>$tempMaxLimit){
                $str = "Your pet appears to have an elevated temperature of: " .$e. " at time "  .$MyTimeArray[$count]."";
                array_push($results, $str);

            } elseif(floatval($e)<$tempMinLimit){
                $str = "Your pet appears to have a low temperature of: " .$e. " at time "  .$MyTimeArray[$count]."";
                array_push($results, $str);

            }
            $count = $count+1;
    }
    return $results;

}
```

Figure 26: TemperatureCheck() function which identifies when the dog's internal temperature is either too high or low and pushes the results to the frontend string array.

```
function checkExistingHealthConcerns( $weight, $age, $MyArray){
    $count = 0;
    // small puppies 220 bpm identify Sinus Tachycardia
    if ($weight < 15 &&  $age < 1){

            foreach($MyArray as $heart){
                if($heart > 220){
                    $count = $count + 1;
                }
            }

    }
    // small dogs 180 bpm identify Sinus Tachycardia
     if ($weight > 15 && $weight < 30 && $age > 1){

            foreach($MyArray as $heart){
                if($heart > 180){
                    $count = $count + 1;
                }
            }

    }
    // standard size 160 bpm identify Sinus Tachycardia
     if ($weight > 30 && $weight < 55 && $age > 1){

            foreach($MyArray as $heart){
                if($heart > 160){
                    $count = $count + 1;
                }
            }

    }

    // giant breeds 140 bpm
    if ($weight > 55 && $weight < 70 && $age > 1){

            foreach($MyArray as $heart){
                if($heart > 140){
                    $count = $count + 1;
                }
            }

        }

     if($count >0){
            return "Flagged: Possibility for Sinus Tachycardia with vet provider. High heartrates appeared " .$count. " times.
                ";
        }
}
```

Figure 27: Function with checks for health concern patterns, specifically Sinus Tachycardia

```
function checkWeightConcerns($dogBreed, $weight, $age){
    $flagWeight = False;
    if ($dogBreed == "Bulldog" || $dogBreed == "Chihuahuas" || $dogBreed == "Maltese" || $dogBreed == "Dachsund" || $dogBreed == "Pomeranian" || $
        dogBreed == "Pug" || $dogBreed == "Boston Terrier" || $dogBreed == "Shih Tzu" || $dogBreed == "Yorkshire Terrier"){
        if ($weight > 40 && $$age >1){
            $flagWeight = True;
            echo "Weight of pet could possibly be too high for their age and breed";
        }

        if ($weight < 15 && $$age >1){
            $flagWeight = True;
            echo "Weight of pet could possibly be too low";
        }

    }
```

Figure 28: Function which checks for health concerns around weight.

The complex algorithm differs in that it works with heart rate ECG signals as highlighted in the algorithm analysis introduction. To train the algorithm, ECG signals were analyzed and split up into a 70% training set and a 30% testing set. The histogram distribution of this PhysioNet data is shown in Figure 29 where there are clearly outliers of abnormal frequencies seen at 1.0 and 0.1.

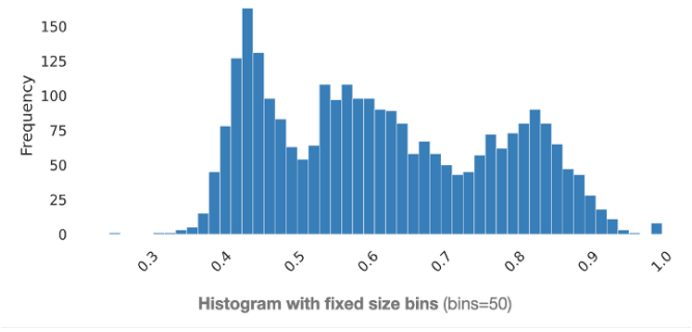

Figure 29: Histogram of PhysioNet data with abnormalities of ECG frequency data seen at 1.0 and 0.0 and 0.3

Figure 30: Figure of data flow of Isolation forest algorithm and the train and test csv datasets used.

This dataset contains 'normal' heartbeats which the algorithm trains upon, and then the outliers are introduced in the test set to validate that the algorithm is detecting abnormal beats per minute. To identify the heart rate abnormalities, the maximum and minimum peak signals are labelled with 1, the rest of the 'normal' rates are given 0 as seen in Figure 31.

```
def train_isolationforest(iforest,train_X,train_y,test_X):
    iforest = IsolationForest(n_estimators=100,contamination=0.05)
    iforest.fit(train_X,train_y)
    test_preds = iforest.predict(test_X)
    test_scores = iforest.decision_function(test_X)
    return test_preds, test_scores

def map_prediction_values(y):
    # if returns -1 if the item is a heartrate that is abnormal
    # otherwise a 0 is returned to signify no cause of concern
    y[y == 1] = 0
    y[y == -1] = 1
    return y
```

Figure 31: Isolation Forest training and mapping of predictions depending if 0 or 1 peaks are returned.

```
X_train, y_train = create_X_y(train_df)
X_test, y_test = create_X_y(test_df)
exp_id = get_experiment_id("heartbeatAnomalityDetection")
mlflow.tensorflow.autolog()
with mlflow.start_run():
    elif params['model_name']=='if':
        from ad_if import *
        iforest = IsolationForest(n_estimators=100,contamination=params['contamination'],random_state=100)
        eval_meas = evaluate_test_if(iforest,X_train,y_train,X_test, y_test,test_df,params)
        print(eval_meas)
        dagshub_log(params,eval_meas)

        mlflow.sklearn.log_model(iforest,artifact_path=MODELS_DIR,registered_model_name=params['mlflow_model_name'
    else:
        print('need to specify the contamination score with the correct if model name')
mlflow.end_run()
```

Figure 32: Splitting of the data into test and train sets using isolation forest and printing of the evaluation measures.

```
def evaluate_isolationforest(model, train_X,train_y,test_X,test_labels,test_df,params):

    predictions, _ = train_isolationforest(model,train_X,train_y,test_X)
    predictions = predictions.astype(int)
    predictions = map_pred_values(predictions)
    test_df['test_predictions'] = predictions

    precision = precision_score(test_labels, predictions)
    recall = recall_score(test_labels, predictions)
    f1 = f1_score(test_labels, predictions)

    # test for the f1, recall, and precision to determine how well isolation forest is
    # with given parameters to detect abnormalities
    logging.log(logging.INFO,
                f"{model.__class__.__name__} model Precision test: {precision}")

    logging.log(logging.INFO,
                f"{model.__class__.__name__} model Recall test: {recall}")

    logging.log(logging.INFO,
                f"{model.__class__.__name__} model f1-score test: {f1}")

    evaluation_report = {'recall':recall,"precision":precision,"f1":f1}

    log_metrics_plot(test_df,params,evaluation_report)

    return evaluation_report
```

Figure 33: Printing function to identify a contamination score of 0.5%, an f1 score of 72%, 64% precision and 85% recall.

To determine if a heart rate is an abnormality, the difference of the current value and previous value of the heart rate is stored in a new column of the database. This helps in detecting any patterns such as the heart consistently beating too rapidly or out of rhythm from the frequency of the minimum and maximum peaks. The data flow is seen above in Figure 32 with the Isolation Forest algorithm being used with the from sklearn.ensemble import IsolationForest package as shown in Figure 34. The only hyperparameter needed is the contamination score which doesn't impact the algorithm and is only a threshold for the amount of outliers introduced [20]. Isolation forest isolates outliers and partitions between all heart rate data [21]. Abnormalities are detected as they have a smaller difference between the partition and the minimum or maximum peaks and will naturally be on the ending nodes in a binary tree

model [15]. This is why the algorithm trains using 'normal,' data first before introducing abnormal heart rate data. Isolation Forest resulted in a contamination score of 0.5%, an f1 score of 72%, 64% precision and 85% recall. In Figure 34 is the code to get these results. Hence, it met the performance requirements of 3.4, 3.5, 3.6, and 3.7. This means there will be some false positives, but it can be adjusted in the user interface section where the user can validate or identify a misdetection. Over time as more data is collected, the recall/accuracy percentage is expected to go up since it will keep retraining on the previous data. Isolation Forest outputs -1 when an abnormality is detected, and a 1 otherwise. This way it can output a string of the abnormalities. The code is very long with many interconnected parts, hence key parts are shown in this section with the GitHub link in the Appendix. To visually show the data, an API link was created and connected to Git to show the true heart rate and the predictions highlighting the abnormalities as seen in Figure 34.

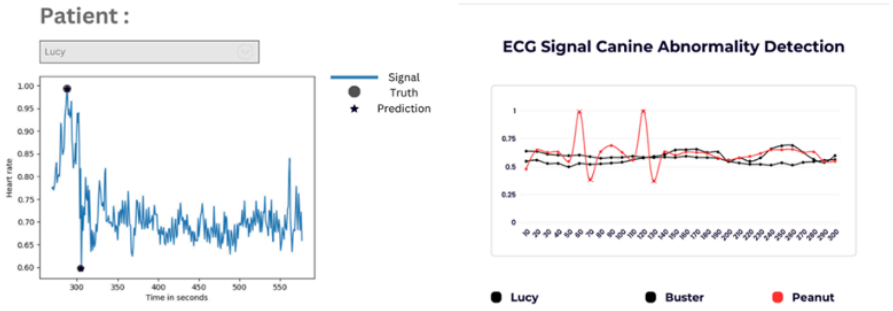

Figure 34: Predictions of abnormal heartbeat identifications visually in graphs.

3.5 - Bill of Materials

The project was well below the initial predictions of $242.36. The list of purchased materials for the project and the total cost of $196.57 can be seen in Table 4. This suggests that mass production would reduce the cost of sensors and allow the builder to purchase higher quality sensors without going over the predicted costs.

Table 4: Full bill of materials with final total cost

Items	Costs ($)
3 Axis Accelerometer Gyroscope (step count)	15.89
High Sensitivity Sound Microphone Sensor	12.69
NodeMCU ESP8266	12.99

2Pcs Heart Rate Pulse Sensor	18.91
Digital Barometric Pressure and Temperature Sensor	13.50
Arduino Uno	39.99
3 Pack Solderless Plug in Breadboard with 3 Pack Jumper Wires	18.99
AA Battery Holder Bundle Arduino Holder	9.99
Dog Harness	31.99
XAMPP (free software to host the website and database)	0
Total (before taxes)	173.94
Total (including taxes)	196.57

4.0 -Testing And Verification

4.1 - Hardware

Each hardware component was verified and met the project standard by being able to produce accurate results and successfully sent to the database. During the simulation stage of testing each sensor individually, a strategy of validating the success of a sensor was to compare it with a device that is able to measure that feature accurately. For instance, the heart rate sensor was first validated by the accuracy of the results given while comparing the result with the Apple Watch heart rate feature. Initially, the sensor was extremely inaccurate due to poor manufacturing, which needed to be resoldered. After inspecting the physical qualities and ensuring the sturdiness of the sensor, the results still presented error readings and spikes when compared to the Apple Watch results. The error was then minimized further by sampling the result each minute, providing heartbeat sensor readings that were within the vicinity of the Apple Watch result. The thermal sensor was also tested and validated by comparing it to the readings from the room thermostat and observing temperature changes and results when place skin contact. In addition, test results were also validated with results from previous experiments. A blog written by Dr.Illyena Hirskyj-Douglas, who specializes in Human and animal computer interaction, was able to successfully test an electrocardiography (ECG) sensor with her dog, which provided an accurate result when the dog was at rest [22].

All sensors were validated using the hardware specifications table on Table 1 when integrated together as well as when storing into the database. In comparison to the project's performance and the hardware table specification, the pet harness can successfully reach the target power consumption of 7 hours, is able to monitor and read heartbeats from the pulse sensor after error reducing, accurately measure the body temperature of the pet, and have a height below the target value of 500g depending on the dog breed. However, measuring and reading from a respiratory sensor was not achieved as it was difficult to measure the accuracy and compare the result with another source or device. The purchased sensor could not be used to adequately measure breathing rates for a dog.

4.2 - API and Database

The API and database were tested with the use of Postman, an application that allows users to send http requests. This software was ideal to isolate the API from any outside factors and create unit tests since the request body was entered by the user. An example of a successful API test with the Postman application is seen in Figure 35. The output is a 200 response code. The test was completed by a visual check of the database tables to ensure that the Heart_Rate, Steps, Temperature, and Audio tables all had updated values. A series of test input featuring different input types and missing values were executed to confirm that the database attributes only accepted the predicted input. For instance, the heart rate must be a number as the database attribute type is set to Decimal(6,2). Gradually, the API was integrated with the Wi-Fi module and the results were confirmed through the updated database tables.

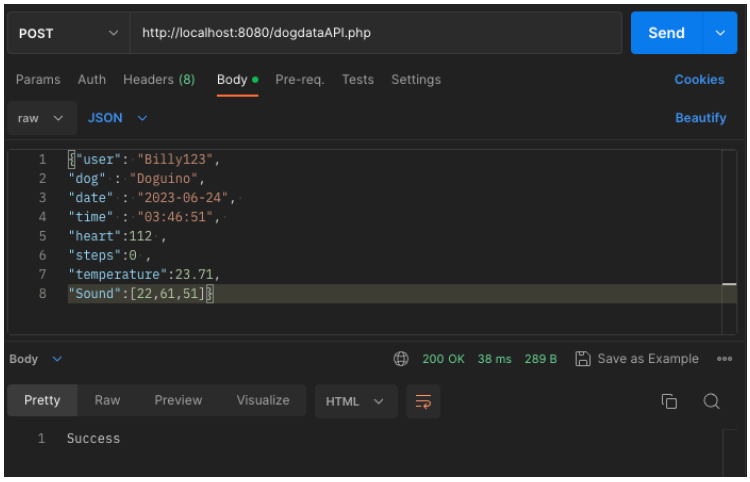

Figure 35: The API test

The POST requests were consistently successful, and the simple API updated the database in real-time. Therefore, the data processing rate was approximately 1 minute including the time it took to collect and record health measurements which allowed the user to access the dog's health information at any time.

4.3 - Algorithm

To test the original algorithm, a test set was fed into the functions. The test set contained a small dataset of heartbeats, their corresponding time array, and a mock user inputted pet detail. The details where the pet's weight, breed, age, sensitivity, and any underlying conditions. By using a test set, the team already knew how many algorithms should be identified and could visually see in the graphs and frontend interface if they all were detected. Once the identification rate was 100%, the team fed in three different test sets. One being that of a diabetic dog with high heartrates (atrial fibrillation), another of a puppy with Sinus Tachycardia (elevated heartbeats occasionally), and lastly a 'normal' test set. This validated that the algorithm was identifying patterns in the data. Lastly, the group tested everything together with the algorithm by using the sensors to read in heartrate data from the teammates into the backend. From this data it was then tested to ensure that there were no flags from the algorithm since the teammates' heartrate readings were normal. For the more complex algorithm using ECG signals, to determine its accuracy, a test set was fed into the algorithm and determined to have an f1 score of 72%, 64% precision and 85% recall. Hence, it met the performance requirements of 3.4, 3.5, 3.6, and 3.7. Although, there will

be some false positives, but it can be adjusted in the user interface section where the user can validate or identify a misdetection. This algorithm would be used when the team invests in a better more accurate heartbeat sensor that wouldn't have trouble reading from a dog with a lot of fur. It wasn't possible to tell if both algorithms were meeting industry standards as all current collars and harnesses on the market do not release information in terms of their insights and prediction accuracy or the amount of false positives [5].

4.4 - Frontend

The frontend code tested a variety of different buttons used to ensure the website was fully functional. Tests included testing buttons from page to page, creating profiles for owners and registering pets to see if it was successfully created in the database. When an expected result did not occur, the team started to debug the code part by part to find the error, fix it and test again. Once a profile was successfully created, filling out the same required information in the login page to determine if the page was successfully able to log in the owner.

For testing purposes, multiple different pets were registered, and data was inserted into the database. On the *Monitoring* page, the different pets were selected from down menu and results shown in the graphs and tables were compared to the data inserted to the database to ensure it is showing the correct data on the page.

5.0 - Stakeholders

The main stakeholders of this project are pets and pet owners. Pet owners have the need to safely monitor their dog's health non-invasively and accurately. Below, Table 5 describes each of the main SSERP points taken into consideration when building a smart pet harness for the stakeholders.

Table 5: Stakeholder factors

Safety Factors	a. Weight of harness with hardware components b. Restrictive movement c. Durability and flexibility d. Battery leakage or electricity flow shocks

Codes/Standards	e. Research on dogs must be done in a Canadian facility that has been certified on the Government and the Ontario Animals for Research Act must be followed
Privacy	f. Canadian Privacy Act to ensure privacy of the database and its security
Manufacturability	g. Simple inexpensive and safe design comes already assembles
Ethics	h. Ethical Guidelines to follow when using Animals for research as well as Engineering Ethical Guidelines to follow to build products ensuring the safety of all living beings and acting in the best possible intentions
Cost	i. Pets are expensive enough as they are. Protype must be cost beneficial.

a. Weight: To safely measure health readings, the group had to design a harness with the needed hardware sensors and wires that wasn't heavy and didn't cause any strain on the pets. The dog shouldn't have to exert any excess strain in having the harness on its back since it could be dangerous and result in incorrect health readings such as a high heart rate from overexertion.

b. Restrict Movement: It was also important to ensure that the product was non-restrictive and didn't impede any of the pet's movement or breathing abilities. The group purchased an affordable harness worth $30 on Amazon from Rabitgoo with over 50 000 positive ratings and guaranteed no pulling or chocking of the pet with two metal rings for walking.

c. Durability and Flexibility: In case the pet wanted to walk around, bite the harness, play with other dogs, and lie down on their back, the harness with the hardware components needed to be flexible and durable. This is why the design has all hardware components tucked away in a small flat box onto the dog's back as seen in Figure 36 below. It has padding all around the hardware components which allows for the dog to roll around and not remove any wires out of the Arduino or break any components. The purchased harness is also adjustable with straps and made of mesh padding that is non-toxic for dogs and unable to be bitten into due to the many nylon fibers interwoven together.

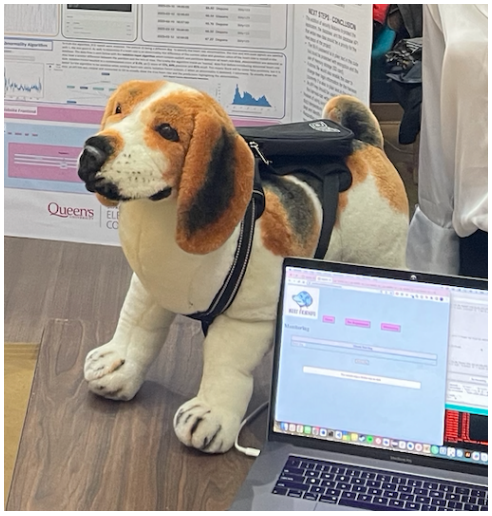

Figure 36: Depiction of the dog harness system on Doguino.

d. Battery Leakage and Electricity Flow Shocks: The hardware components were packaged in a little padded box onto of the harness. This way nothing is directly touching the dog. The box surrounding the hardware components was made of non-conductive material and was double insulated so that any leakage isn't possible to seep out.

e. Codes/Standards: Since the projects main stakeholder is animals that cannot consent for themselves, it results in a lot of codes/standards specifically for testing on dogs that must be followed as outlined by the government of Canada [23]. Research facilities must be certified in Ontario and the Ontario Animals for Research Act outlines the many details around animal care and their treatment [24]. Since the team didn't have access to any dogs with underlying health conditions and due to the amount of time it would have gotten to receive an approval to research on them, the group skipped this component and tested the sensors on themselves. In the future to ensure animals' safety, the team would go to a certified animal research center and obtain heartrate, respiration, and temperature data.

f. Privacy: Since the application is storing private information about the pet and the pet owner, The Canadian Privacy Act must be followed [25]. Before use of the product, a data agreement waver would be signed by the pet owner where all the information about the dog will be disclosed

(heartrate, temperature, weight, breed) and how it will be used by the algorithm. They own all rights to their data though and the database will be protected with hash tables and encryption for data integrity and protection.

g. Manufacturability: Before manufacturing, the team would do many validation tests with dogs as many unpredictable data results or actions could occur. They will keep in mind the codes/standards and all the safety components they have built into the harness. Pet owners will love the inexpensive aspect of the pet monitoring harness and how it comes already assembled with the hardware box having the ability to be removed or not.

h. Ethical: The team acted with the best possible intentions and kept their love for dogs in mind when creating the final design. They too have pets that they want to be safe and by creating the prototype with very specific safety components built in they are ensuring they have no intention of ever hurting the dog and instead providing readings that may help keep them healthy. Since there are many legal laws and standards around introducing health gadgets into the market and the possibility of giving false positives, the pet owners would be given waivers to sign and agree to consult a medical veterinary professional and only use the monitoring harness as an additional source of data about their dog.

i. Cost: Pets are already very expensive and if they have health concerns and frequent visits to the vet, pet owners would love to monitor their dog's health in a more economical way. Since the product is very inexpensive, with mass manufacturing production it will be even more inexpensive while providing accurate real time health readings.

6.0 - Compliance with Specifications

Table 6: Hardware System Specifications with achieved value

Specifications		Target value	Tolerance	Achieved Value
1	Overall harness weight including the sensors, fabric, and Arduino	*500 g*	*+ 500 g*	*150 g*
2	Reliable WIFI connection	Constant connection	Loss of connection below 1 min	Constant connection
3	Data processing rates with minimal delays including the time it takes for the sensor data to be saved in a database by the Arduino	Maximum 5 min delays	+ 1 min	No delays
4	Low Power Consumption resulting in a long battery life	Battery lifespan of 7+ hours	- 3 hours	Battery lifespan of 7+ hours
5	The heartbeat monitoring sensor must detect all heartbeats per minute in the range from 50 to 130 heartbeats/min.	All heartbeats/min	+/- 10 heartbeats per min	All heartbeats/ min
6	The respiratory sensor must detect respiration per minute in the range of 5 to 35 breaths/min.	All breaths/min	+/- 5 breaths per min	Not Achieved
7	The temperature sensor must accurately measure body temperature in the range from 99°F to 104°F.	Current temperature with decimal values.	+/- 1°F	Current temperature with decimal values.

As seen in Table 6, the hardware specifications were all met except for number 6. This is because the team didn't end up measuring respiration as it would apply a lot of pressure to the pet. Different factors such as exposed hair and skin and positioning of the sensors to identify when the diaphragm inhales and exhales were too tricky to measure accurately with our prototype. Our final design does meet the hardware specifications of being under 500g and is 150g specifically so that a small dog can even carry the harness components on its back. We could even reduce the weight further by using lighter batteries such as 9A batteries since they make up most of the weight. As well there is reliable Wi-Fi connection and no delays in response to saving data from the Arduino. The battery lasted all throughout our project meeting the requirements of having a lifespan greater than a day. Also, the temperature and

heartbeats sensor effectively monitor and report readings. Due to the safety concerns of testing on dogs, we had to validate our sensors using ourselves and cross-reference with Apple Watches and thermostats.

Table 7: Software System Specifications with specification met

1	**Functional requirements**	**Specification Met?**
1.1	Application to give the user access to data and settings	Yes
1.2	Algorithm to detect common heart rate anomalies (e.g., arrythmia) in real-time	Yes
1.3	Algorithm to detect common respiratory rate anomalies (e.g., obesity) in real-time	No
1.4	Algorithm to discover sudden spikes in data in real-time	Yes
1.5	Algorithms adapts to user feedback (Users are notified of abnormalities and can approve or disapprove of a detection)	Yes
1.6	Application considers users preference such as abnormality sensitivity levels or type of abnormalities detected	Yes
2	**Interface requirements**	
2.1	Data page to display the temperature, respiratory, movement, and heart rate data in the form of graphs	Yes
2.2	Notification page to display notifications of abnormality detections	Yes
2.3	Settings page to adjust dog type, age, weight, size, name, and user preferences	Yes
3	**Performance requirements**	
3.1	Retrieval of data from a database is under 2 minutes	Yes
3.2	Change of the application page is under 10 seconds	Yes
3.3	Real-time notifications are based on algorithm results	Yes
3.4	After a training period, the chosen abnormality algorithm should aim to reduce the rate of false detection under 30%.	Yes
3.5	The number of missed detections (the abnormality is never detected) should be extremely low. The user should be able to set the sensitivity level of the device to reduce or increase the number of notifications received.	Yes
3.6	Performance will be measured by comparing the link between false detection and missed detections. The algorithms that reduce the number of missed detections while keeping the number of false detections low will be chosen to maximize the probability of correct detection.	Yes
3.7	Algorithms will use parameters such as healthy heart rate, respiration frequency, and body temperature to find abnormalities. Algorithms will use unsupervised learning to detect any extremes in data and will use models of common health issues.	Yes

The software specifications table outlined in Table 7 also only had one functional requirement not met, being an algorithm for common respiratory rate anomalies (functional requirement 1.3). The same reasoning applies that we didn't measure respiration in the end. The Abnormality Analysis section of

Implementation outlines how functional requirements 1.1, 1.2, 1.4, 1.5, and 1.6 were developed. The interface requirements are all met and outlines in Frontend Analysis section of Implementation. Lastly, the performance requirements are met since database retrieval and application page loading is immediate with no delays. The algorithm results update in real-time as more data is collected from the sensors. The Abnormality Analysis section also outlines how performance requirements 3.4, 3.5, 3.6, and 3.7 are met where the user can change the sensitivity levels of the algorithm and the recall rate ended up being 85%.

7.0 - Conclusions & Recommendations

Although a complete project was produced and working, there were many challenges faced when creating and implementing it. To start off, when building the smart harness, a goal put in place was to find cost efficient parts with fast delivery dates. This was to ensure that there was ample time to build the harness. As a result, the components were poorly assembled by the manufacturer and was not reading the correct readings. Some parts came in a pack of two and often only one of the two worked a sufficient amount. This caused a delay in assembling the harness as pieces needed to be replaced and re assembled, as well some parts needed to be soldered together for it to work. Another hurdle that we avoided when building the harness was potentially assembling the circuit onto a PCB. The main concern was that there was only one working prototype and with the lack of good parts a risk was not taken to assemble the circuit on the PCB. A real work issue that came in concern with the harness was that the hardware components are sensitive and with an active pet it may get damaged and produce false results.

When trying to tackle and test each sensor, a realization was made that were a numerous amount health concerns but each sensors capabilities were only to a certain extent. As a result, with the level of pet information obtained from the owner the algorithms analysis is limited. In this same scope, through research we discovered a large number of breeds that could possibly be at risk to medical conditions but there is also a large number of breeds in general. The website asks the owner the pets breed, size, weight and age and any other factors, but because of the large number of breeds it would be best to group similar breeds together or narrow the focus of the product to the average breed and health. For further

improvement of the algorithm, when looking that the analysis of the pets' tempera ture, the algorithm could potentially go more in depth and identify complex patterns. As well, asking the owner if their pet has any pre-existing conditions will help modify the algorithm to consider a multitude of dog types predisposed to certain health complications so that the detection of common breed issues could be achieved.

Recommendations to further improve this product is to run more simulation tests on the hardware components to ensure they are working as needed. As well, ordering products much earlier to ensure hardware components are more reliable and have higher accuracy. An issue that occurred at the end of creating this project was the lack of time had to merge both the hardware components and software components. As problems occurred during the merging process debugging and solving each issue took longer than expected.

The addition of security features to protect all components of the product is a necessary factor to include, especially to protect the website application, the database and the database API that writes new data. The Wi-Fi password is in plain text in the code but could be protected with encryption and the use of memory storage (SD card). A stored file would also enable the user to change their login information for the harness (currently it is also in plain text in the code). An API key should be used to ensure that only the harness can write to the database.

Respiration could not be accomplished at this stage of the project as it could not be monitored as it would apply too much pressure to the pet. Researching more ways to effectively do this without harming the pet would need to be done to analyze the performance and feasibility of measuring respiration safely. As well, for legal and ethical reasons testing could not be done on any animals. Once a prototype is fully completed and tests have been completed to verify that the prototype will not harm any animal, testing the *Best Friends* harness on animals can begin. Upon all tests being passed, the *Best Friends* harness can be put on sale for the public.

8.0 - Overall Effort

Table 8: The overall team effort

Name	**Overall Effort Expended (%)**
Ali Irhouma	100
Courtney Orcutt	100
Talia Edwards	100
Vanusha Vicknesvaran	100

References

[1] I. Richardson, "SpringerLink," 21 August 2017. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-61222-5_9#Bib1. [Accessed 10 April 2023].

[2] A. Staff, "American Veterinary Medical Assocation," 2023. [Online]. Available: https://www.avma.org/resources-tools/pet-owners/petcare/microchips-reunite-pets-families/microchipping-faq. [Accessed 10 April 2023].

[3] "PetPace Monitor Your Pets Health," PetPace, [Online]. Available: https://preorder.petpace.com/. [Accessed 5 February 2023].

[4] VetMeasure, [Online]. Available: https://vetmeasure.com/shop/. [Accessed 5 February 2023].

[5] C. Mauran, "Biometric dog collars claim to track your dog's vitals. But are they fur real?," Mashable, 23 January 2022. [Online]. Available: https://mashable.com/article/are-wearable-health-trackers-for-dogs-reliable-experts-weigh-in.

[6] "How Much Does It Cost to Own a Dog in Canada?," PHI Direct Pet Health Insurance, [Online]. Available: https://www.phidirect.com/blog/the-average-cost-of-owning-a-dog-in-canada. [Accessed 11 March 2023].

[7] J. L. Rowell and C. E. Alvarez, "Dog models of naturally occurring cancer.," *Trends in molecular medicine,* vol. 17, no. 7, pp. 380-388, 2011.

[8] K. Grüntzig, R. Graf, G. Boo, F. Guscetti, F. Hässig, K. Axhausen and A. Pospischil, "Swiss canine cancer registry 1955–2008: occurrence of the most common tumour diagnoses and influence of age, breed, body size, sex and neutering status on tumour development," *Journal of comparative pathology,* vol. 155, no. 2-3, pp. 156-170, 2016.

[9] A. Helwatkar, D. Riodan and J. Walsh, "Sensor Technology For Animal Health Monitoring," International Journal on Smart Sensing and Intelligent Systems, 2014.

[10] "Detect, Measure & Plot Heart Rate using Pulse Sensor & Arduino," [Online]. Available: https://lastminuteengineers.com/pulse-sensor-arduino-tutorial/.

[11] "Interface MPU6050 Accelerometer & Gyroscope Sensor with Arduino," 29 December 2020. [Online]. Available: https://lastminuteengineers.com/mpu6050-accel-gyro-arduino-tutorial/.

[12] "Instructables," [Online].

[13] "Interface BMP180 Barometric Pressure & Temperature Sensor with Arduino," 29 November 2019. [Online]. Available: https://lastminuteengineers.com/bmp180-arduino-tutorial/.

[14] "Interface Sound Sensor with Arduino and Control Devices With a Clap," 29 November 2019. [Online]. Available: https://lastminuteengineers.com/sound-sensor-arduino-tutorial/.

[15] A. Sološenko, A. Petrėnas, B. Paliakaitė, V. Marozas and L. Sörnmo, "Model for Simulating ECG and PPG Signals with Arrhythmia Episodes," *PhysioNet,* vol. Version 1.3.1, 2022.

[16] A. Wajdan, T. Jahren, M. Villegas-Martinez, M. Krogh, F. H. Khan, A. Espinoza, P. S. Halvorsen, H. H. Odland, O. J. Elle and E. W. Remme, "Epicardially attached cardiac accelerometer data from canines and porcines," *PhysioNet,* vol. Version 1.0.0, 2023.

[17] D. Primovic, "Normal vitals for a dog," Burlington Emergency & Veterinary Specialists, 30 September 2015. [Online]. Available: https://bevsvt.com/normal-vitals-for-a-dog/#:~:text=For%20dogs%2C%20a%20normal%20heartbeat,slower%20the%20normal%20heart%20rate..

[18] L. Simon and H. Hollinger, "Irregular Heartbeat in Dogs," Wag!, 27 September 2021. [Online]. Available: https://wagwalking.com/condition/irregular-heartbeat.

[19] M. Weir and L. Buzhardt, "Taking Your Pet's Temperature," VCA animal hospitals, 2022. [Online]. Available: https://vcahospitals.com/know-your-pet/taking-your-pets-temperature.

[20] "Exercise: Anomaly Detection," Machine Learning for Scientists, 2020. [Online]. Available: https://ml-lectures.org/docs/unsupervised_learning/Anomaly_Detection_RNN_AE_VAE.html.

[21] K. Dhiraj, "Anomaly Detection Using Isolation Forest in Python," Paperspace Blog, 09 April 2021. [Online]. Available: https://blog.paperspace.com/anomaly-detection-isolation-forest/#:~:text=machine%20learning%20is.-,What%20Is%20Isolation%20Forest%3F,on%20the%20Decision%20Tree%20algorithm.

[22] I. Hirskyj-Douglas, "Doguino: Using Arduino to Measure a Dogs Heartbeat," [Online]. Available: https://ilyena.com/2017/02/17/doguino-using-arduino-to-measure-a-dogs-heartbeat/. [Accessed 10 April 2023].

[23] Animal Alliance of Canada, "No Pets In Research," Animal Alliance of Canada, 2023. [Online]. Available: https://www.animalalliance.ca/campaigns/pets-research/.

[24] Government of Ontario, "Animals for Research Act," Government of Ontario, 2019. [Online]. Available: https://www.ontario.ca/laws/statute/90a22.

[25] Government of Canada, "Access to Information and Privacy," Government of Canada, 07 10 2022. [Online]. Available: https://www.canada.ca/en/parole-board/corporate/transparency/access-to-information-and-privacy.html.

[26] PETMD, "Rapid Heart Rate in Dogs," 2022. [Online]. Available: https://www.petmd.com/dog/conditions/cardiovascular/c_dg_rapid_heart_beat.

[27] PETMD, "Irregular Heartbeat in Dogs," 2022. [Online]. Available: https://www.petmd.com/dog/conditions/cardiovascular/c_dg_arrhythmia.

[28] Cornell University College of Veterinary Medicine, "Arrhythmias (Abnormal Rhythms) in Dogs," 2016. [Online]. Available: https://www.vet.cornell.edu/hospitals/companion-animal-hospital/cardiology/arrhythmias-abnormal-rhythms-dogs.

[29] Petcardia Veterinary Cardiology, "Atrioventricular (AV) Block," 2021. [Online]. Available: https://www.petcardia.com/avblock.html.

[30] PETMD, "Heart Beat Problems (Premature Complexes) in Dogs," 2022. [Online]. Available: https://www.petmd.com/dog/conditions/cardiovascular/c_dg_atrial_premature_complexes.

[31] PETMD, "Breathing Difficulties in Dogs," 2022. [Online]. Available: https://www.petmd.com/dog/conditions/respiratory/c_dg_dyspnea_tachypnea_panting.

[32] Cummings School of Veterinary Medicine, "DIFFICULTY BREATHING (Dyspnea)," Tufts, 2022. [Online]. Available: https://heartsmart.vet.tufts.edu/difficulty-breathing-dyspnea/.

[33] "PetPace Monitoring your Pet's Health," PetPace, [Online]. Available: https://petpace.com/common-dog-diseases/. [Accessed 10 April 2023].

[34] "Detect, Measure & Plot Heart Rate using Pulse Sensor & Arduino," Last-Minute Engineers, [Online]. Available: https://lastminuteengineers.com/pulse-sensor-arduino-tutorial/. [Accessed 10 April 2023].

Appendices

The following appendices contain information about health abnormality parameters, validation testing, and planification.

Appendix A - Health Abnormality Parameters

The appendix defines the parameters that will be detected by sensors and used by the abnormality detection algorithms. The definition of a dog's normal health will be used to identify dangerous data spikes whereas common heart rate issues and respiratory rate issues will be used to design models of possible abnormalities.

Definition of a Dog's Normal Health

Table 9 describes the normal vital rates of dogs. Note that the larger a dog's size, the slower the dog's heartbeat rate will become [17]. Moreover, a pet's body temperature becomes dangerous when it reaches 104°F (40.0°C) or decreases to 99°F (37.2°C) [19].

Table 9: The definition of normal vitals for dogs [17] [19]

Vital Monitoring	Small Dogs and Puppies	Large Dogs (over 30 pounds)
Normal Heart Rate	120 to 160 beats/minute	60 to 120 beats/minute
Normal Respiratory Rate	10 to 30 breaths/minute	
Normal Panting Rate	200 pants/minute	
Normal Body Temperature	101.0 to 102.5°F (38.3 to 39.2°C)	

Common Heart Rate Issues

Table 10 explains common irregular heartbeat symptoms in dogs [18].

Table 10: Common Irregular Heartbeat Issues [18], [26], [27], [28], [29], [30]

Irregular heartbeats	Puppies	Small dogs	Standard Size	Giant Breeds
Sinus Tachycardia: Heartbeat with impulse at a faster rate	> 220 bpm	> 180 bpm	> 160 bpm	> 140 bpm
Sinus arrhythmia	When inhalation occurs, the heartbeat increases. When exhalation occurs, the heartbeat slows down.			
Atrial fibrillation	The heart will beat at a higher-than-normal rate.			

Atrioventricular block	The heart will beat at a slower-than-normal rate.
Premature beats	The heart beats before the normal timing (prematurely).

Some breeds of dogs are predisposed to irregular heartbeats such as Bulldogs, Pug, Pekingese, Saint Bernards, Great Danes, Mastiffs, Boxers, and Doberman Pinschers [18]. Heartbeat issues will cause increased or decreased heart rates [18]. Variance of the heart rate or rhythm in dogs is classified as canine arrhythmia [18].

Common Respiratory Rate Issues

Common breathing issues include labored breathing (dyspnea), rapid breathing (tachypnea), and abnormal panting [31]. Dogs with dyspnea will have difficulty breathing due to heart issues and will have faster than normal breathing rates when at rest [32]. Symptoms may involve coughing and exhibiting visible efforts to breath (belly wall motion) [32]. Rapid breathing and abnormal panting can be detected by comparing the breathing rates to the normal rates in Table 9.

Common Health Issues Algorithm Input and Output

The main health problems many dog breeds face are obesity, arrythmia, arthritis, dehydration and lung infections. Depending on the breed and weight (x input) of the dog, they may be in the obesity category. The dog will breathe faster affecting their respiratory rate (y output) since they cannot get oxygen fast enough. As well, depending on the age (x input) of the dog, their movement trends may change with their dog leg gate being slower (y output). Lastly, depending on the breed (x input), certain dogs will be more predisposition to epilepsy, infection, exercise intolerance collapse (EIC), and Addison's disease.

Appendix B- Progress Planning and Idea Generating

Figure 37 shows the idea generation process.

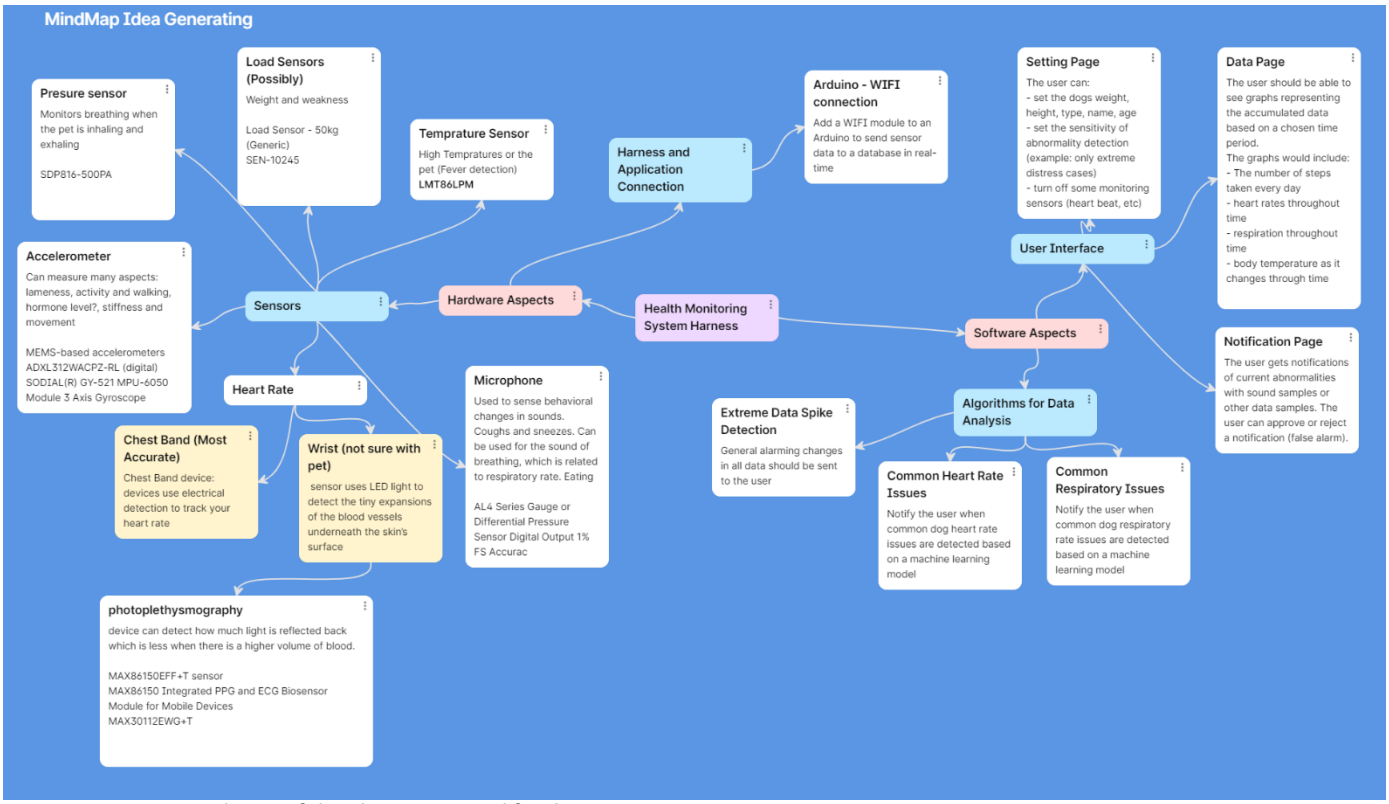

Figure 37: Mind Map of the Ideas Generated for the project