**Boston University**

**Electrical & Computer Engineering**

**EC463 Senior Design Project**

First Semester Report

NoiseHub

Submitted to

Michael Hirsch

Boston University

8 St. Mary’s St

(617) 353-2811

mhirsch@bu.edu

by

Team 8

Team NoiseHub

Team Members

Allen Zou [allenzou@bu.edu](mailto:allenzou@bu.edu)

Benjamin Brewer [bbrewer1@bu.edu](mailto:bbrewer1@bu.edu)

Ibrahim Chand [ichand@bu.edu](mailto:ichand@bu.edu)

Alexander Prior [aprior@bu.edu](mailto:aprior@bu.edu)

Submitted: 12/12/21

#### Table of Contents

[**Executive Summary**](#_heading=h.trf1caiijfif) **3**

[**Introduction**](#_heading=h.1fob9te) **4**

[**Concept Development**](#_heading=h.3znysh7) **6**

[**System Description**](#_heading=h.tyjcwt) **8**

[**First Semester Progress**](#_heading=h.3dy6vkm) **11**

[**Technical Plan**](#_heading=h.1t3h5sf) **13**

[**Budget Estimate**](#_heading=h.4d34og8) **15**

[**Attachments**](#_heading=h.2s8eyo1) **16**

[**Appendix 1 – Engineering Requirements**](#_heading=h.17dp8vu) **16**

[**Appendix 2 – Gantt Chart**](#_heading=h.3rdcrjn) **17**

# 

# Executive Summary

NoiseHub

Team 8 – Team NoiseHub

The problem NoiseHub aims to solve is one all college students experience: finding a suitable study space on campus can be challenging and consume precious time better spent studying. NoiseHub intends to solve this problem by improving efficiency in students' search for study spaces by providing accurate and real-time information on study spaces across campus. This information includes room temperature, noise levels in the room, and a rough headcount to estimate current occupancy.

As a final deliverable, the NoiseHub team will present a fully functioning sensorsuite that will be able to accurately estimate noise level, temperature, and headcount in a room. There will also be a fully functional companion mobile application with user authentication, live, historic, and predictive data trends. This will be achieved using Garmin-Lidar distance sensors to count students entry and exit, thermistors to obtain temperature, and microphones to measure noise level, all contained in the sensorsuite. These sensors will be connected to a Raspberry Pi 4, which will send the data to AWS services for processing. The companion mobile app will query data directly from AWS to give students access to the current room conditions.

The companion app will be tailored to each individual user. After account creation and authentication, the user will be given a survey to determine which room aspects they find important, and the suggested study spaces will be filtered based on the users preferences. Users can also filter study spaces by specific characteristic rankings so they see the most relevant information first.

# Introduction

College students often struggle to find a work space that’s right for them. Some study spaces are too crowded, while others are far too hot. When students can’t find a place to work effectively, they waste precious time hopping between study spaces, and not working at their full potential. Students deserve a system that informs them about the current status of study spaces, so they can spend their time efficiently. The three key features that students tend to care about in study spaces are crowdedness, noise level, and room temperature. Therefore, these are the three measurements we’ve chosen to implement into our project.

Our project is an integrated system with a companion app. The controllers of the hardware unit will be Raspberry Pi’s with attached sensors installed in study spaces across the campus. The attached sensors are three fold. First, the Garmin Lidar V4s, a range finder which uses pulses of light waves to measure distance from the sensor. Two of these will be used in conjunction to measure how many students walk in and out of a given study space. Second, a thermistor which contains a temperature dependent resistor. After measuring the output voltage from the thermistor, it can be converted into degrees Fahrenheit or Celsius. Finally, a usb connected microphone from which we’ll measure the raw decibel level of the room.

Once collected by the Raspberry Pi, the sensors’ data are pushed from the Raspberry Pi to AWS cloud services. In short, these services allow easy data encryption, user authentication, data storage, and cloud computing while the servers are managed behind the scenes by Amazon. Once the data is securely in AWS, students can check the NoiseHub companion app on their mobile devices. The app will query current data from study spaces across campus, and from there, students can see the real-time conditions of them (Figure 1).

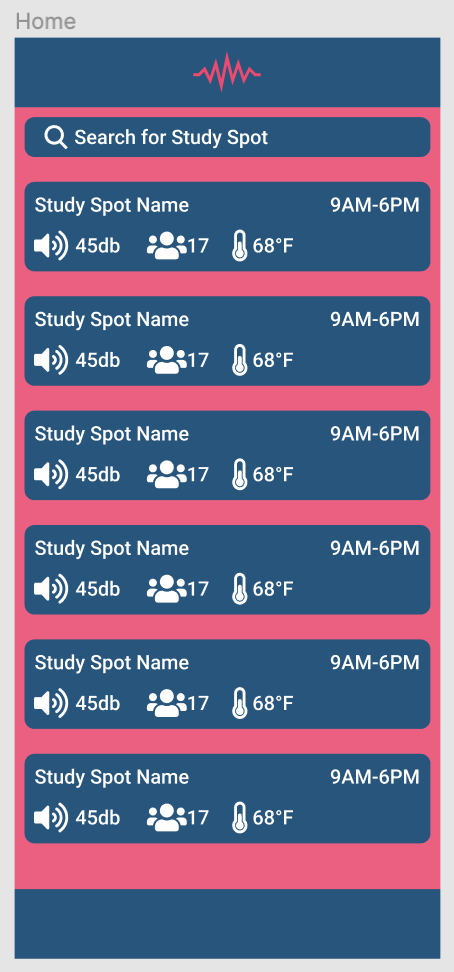


Figure 1. The page in NoiseHub’s companion app where users can view study spaces near them

Upon clicking the tile for a given study space, students can see more in depth information about it, including current data, historical trends, predicted volume and crowdedness peaks, and amenities. Students have the option to sort study spaces near them based on crowdedness, noise levels, and temperature all in increasing order or decreasing order. Furthermore, using stored user preferences, the NoiseHub app will even recommend study spaces best suited for an individual user.

By giving students easy access to real-time study space information, they can save time by picking a study space ahead of time instead of walking around campus to find the perfect one, or settling for one where they’ll have decreased productivity.

# Concept Development

The core issue NoiseHub aims to solve is the inability of individuals to find productive and comfortable places to study. The core concept behind our solution is to provide a service that presents information in an intuitive format to aid individuals when searching for a suitable study space.

The first step towards implementing this solution was to supply characteristics of study spaces that determine their quality. More specifically, what are the variables that students take into account deciding if a location is suitable for work. While many characteristics of study spaces are qualitative and cannot reliably be measured, our team identified three key metrics that can be used by individuals to determine the quality of a space.

The first metric is noise levels. From a student’s perspective, noise is often a critical feature of any study space as it can be extremely distracting if there is loud music or many voices. Inversely, some students prefer a space that is lively to study or have a meeting in without fear of being a nuisance. From an engineering perspective, measuring noise levels is a relatively simple task, requiring inexpensive and readily available microphone components. Using data from the sensor, users can be presented with the decibel level of a space, easing the process of finding a space with appropriate noise levels. Placing microphones in spaces, however, raises concerns over privacy. While our sensors will not record any audio and only gather decibel level data, more research and development will be done to alleviate these concerns.

The second metric is temperature. While this factor is not as noticeable as noise, it can be critical to how comfortable a space is. A study space without air conditioning in the summer is unbearable and a space without heating in the winter is the same. Similar to measuring noise, measuring temperature is a simple task requiring temperature sensors from which data can be gathered to present users with the temperature of a space.

The third metric is headcount. On a busy college campus, all other factors of a study space are irrelevant if there is simply no seat available. The process of going from location to location to find a place to study is frustrating and highly inefficient. The ability to check how busy a space is would be one of the most valuable features in solving the core issue of finding a place to study. From an engineering perspective, however, measuring headcount is the most complex. There are several different methods of measuring the number of individuals in a room, many of which are still undergoing active research. Each offers various tradeoffs in terms of accuracy, privacy, and cost, with no ideal solution. When it comes to this use case, the exact count of individuals in a space is not necessary, rather a general measurement will more than suffice. Privacy and cost, on the other hand, had more strict limitations which made Lidar sensors the most appropriate solution. Alternative options included cameras, thermal cameras, and WiFi or MAC address tracking. Cameras are an inexpensive solution, but increase software complexity and raise significant privacy concerns. Thermal cameras are expensive and unreliable when it comes to accuracy. Tracking WiFi and MAC addresses makes it difficult to retrofit existing spaces and is also inaccurate. Lidar offers the best set of tradeoffs as the sensors are not very expensive or difficult to install, do not raise privacy concerns or increase software complexity significantly, and offer an acceptable degree of accuracy.

Since all three of those metrics require sensors, the solution concept expanded to include a physical device installed in each study space with each of the sensors connected to monitor the space. Target locations come in a wide variety of sizes and shapes, so the device must be modular to support several of the same sensors to distribute throughout a space for more accurate measurements. The device itself must have a small footprint and be simple to install to ensure it can be retrofitted in a majority of locations. The data collected would then be streamed to a central database from which it could be consumed.

With the measurement methodology and physical device concept established, the remaining portion of the core concept is how the information will be presented to the user. Since the target population segment is college students and the service is meant to be used more as a tool, a mobile application is the ideal method. Users can use their smartphones to quickly check current and historical data of various study spaces using the application and efficiently select one. Users are required to register with the service so that it can be tailored to their specific situation and personalize their experience. To increase accuracy as well, users can provide feedback in regards to how busy a space is which is incorporated into headcount measurements and keep errors from propagating.

The only alternative to a mobile application that was considered was a website that would offer the exact same functionality. While a website would avoid issues of cross-platform support or expand the target population beyond those with smartphones, the user experience would be less conducive to the goal of helping students efficiently find a place to study. When trying to find information quickly, most individuals, especially young college students, will reach for their smartphones first rather than using a computer, especially if they’re on the move. Rather than developing a website designed to support both desktop and mobile platforms, sole focus can be given to mobile platforms to develop a more expansive and dedicated solution.

# System Description

NoiseHub’s solution provides an embedded sensor suite that revolves around a Raspberry Pi microprocessor and AWS. As mentioned in the previous section and seen in Figure 2, the system will capture real time data through a decibel sensor, digital thermometer, and a pair of lidar sensors. The data is translated into a human usable format and sent from the Raspberry Pi to AWS where it is stored in a time series database called AWS Timestream [Figure 3]. Then, with user data stored in AWS Cognito and DynamoDB, a curated display of study space information is sent to the user on the React Native application.

|  |
| --- |
| Figure 2. Sensor System Block Diagram |

|  |
| --- |
| Figure 3. Mobile Application Stack Block Diagram |

NoiseHub measures a study space’s ambient temperature using the DS18B20 Digital Temperature Sensor. A digital thermometer was chosen over traditional thermistors since they allow us to read data as a digital signal rather than as an analog signal. The Raspberry Pi does not natively support the ability to read analog data and would require a Raspberry Pi analog digital converter (ADC) hat or secondary ADC and microcontroller. Both solutions would add unnecessary bulk to our constrained housing compartment so a simple digital thermometer with an internal ADC provided us with the simplest and compact solution. The digital temperature sensor is connected to the Raspberry Pi with 3.3V, ground, and a GPIO pin. The temperature is written to a bus device slave file and the microprocessor can be read using a python workload function.

For measuring an estimated headcount, NoiseHub uses a pair of Garmin LIDAR-Lite Optical Distance LED Sensor - V4 to create a lidar tripwire at entrances to a room. The two lidar sensors will be placed parallel to one another and the order by which a person walks into the sight of the lidar sensors will determine if they are entering or exiting the study space. This pseudo-lidar sensor actually uses leds but its reduced price from traditional lidar systems and ability to still accurately read distances made it the perfect fit for this project. It reads accurately between 5 cm to 10 meters and the span of a doorframe, even a large one, falls within that range. The sensor is powered by a 5V power supply from the Raspberry Pi and sends data through the respective I2C ports. To read from the device, a series of register instructions must be completed to prompt a distance acquisition.

The last sensor on our system is the decibel sensor. Although it has not been integrated with the microprocessor yet, we will tentatively be using the Adafruit Mini USB Microphone Model 3367. Similar to the temperature sensor, the Raspberry Pi will use the microphone to sample noise levels periodically. We chose this device because other microphone chips encounter the same issue as traditional thermistors, as they require an analog port on the microprocessor. Adafruit’s Mini USB Microphone, on the other hand, has much greater plug and play capability and is extremely compact so it will be minimally invasive in the limited capacity of our system’s enclosure.

After a sensor has been sampled, the data is packaged into a JSON payload that is sent over MQTT to AWS IoT Core [Figure 4]. IoT Core serves as the endpoint for our AWS data ingestion. The Raspberry Pi is able to build an encrypted TLS tunnel to AWS via IoT Core. Once data is ingested into IoT Core, a rule forwards the data to AWS Timestream for temporal data storage.

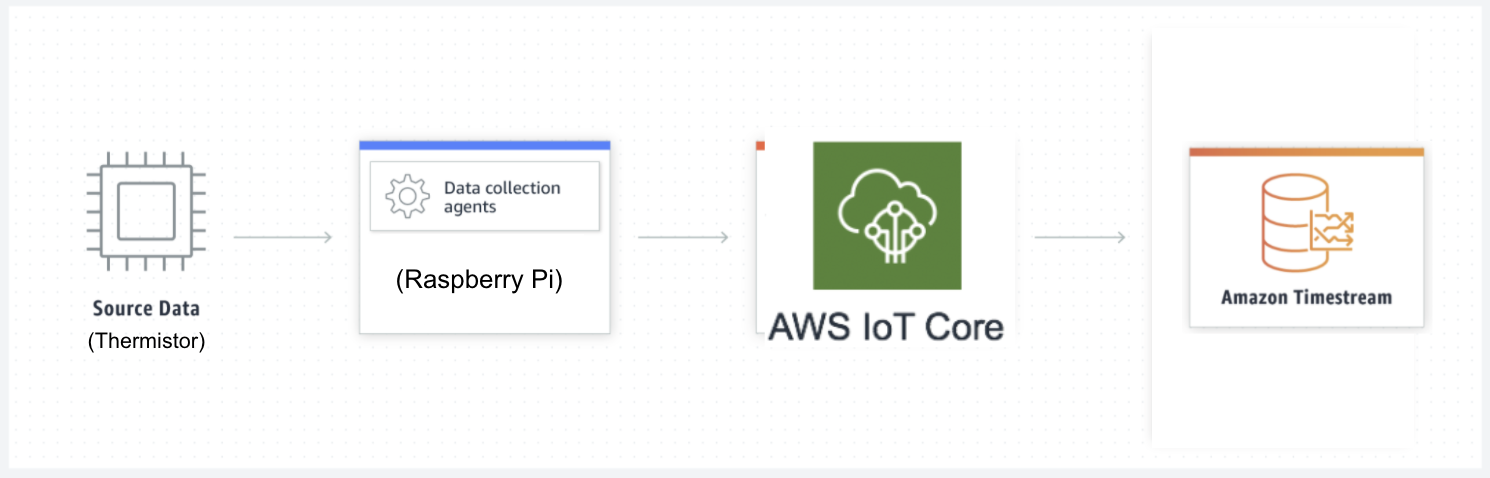


Figure 4. AWS Dataflow

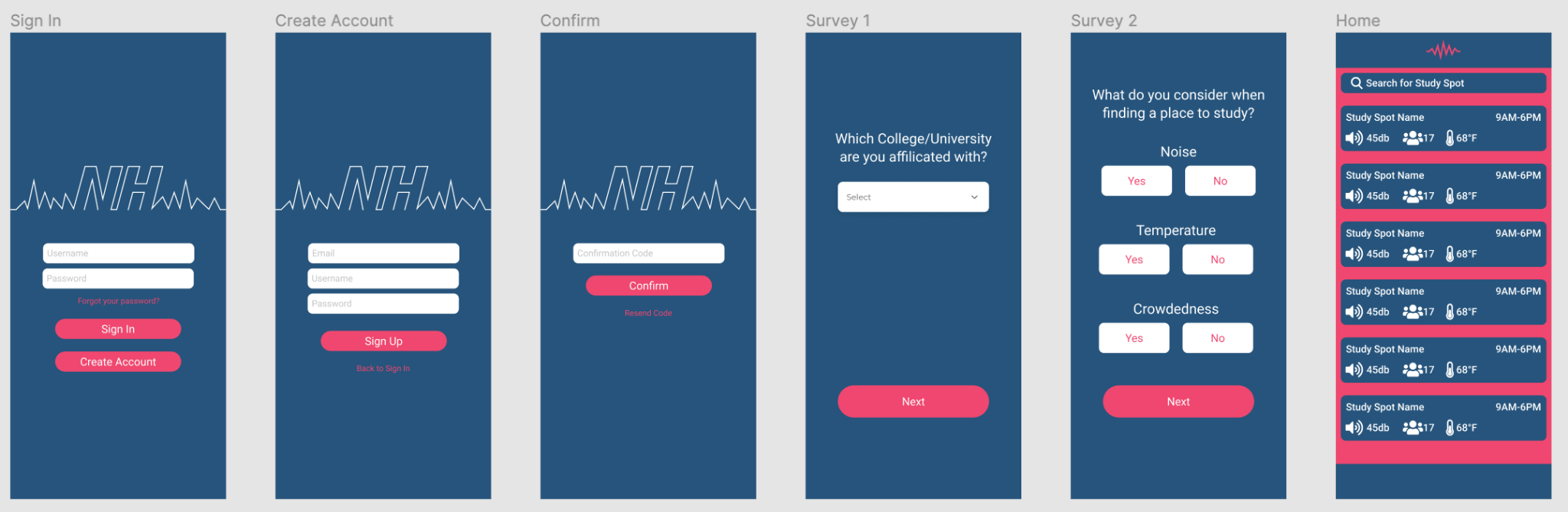


Figure 5. Mobile Application UI/UX

# First Semester Progress

This semester, our team had the ultimate goal of developing a functional prototype consisting of at least one sensor actively gathering data and transmitting it to a backend from which a mobile application would receive said data and display it. In order to accelerate development, a two-pronged approach was used, dividing into hardware and software.

In terms of hardware, our team acquired thermistors, lidar sensors, and microphones along with a Raspberry Pi. Primary focus was placed on enabling the lidar sensors due to their significant complexity relative to the other sensors. This semester, we were able to successfully connect the lidar and thermistor sensors to the Raspberry Pi and receive data from them. During our First Deliverable Testing, we transmitted data from the sensors to the Raspberry Pi for 10 seconds and demonstrated the outputs were accurately and reliably measured. Looking at Figure 6, the “Temperature” column holds data from the thermistor and indicates the temperature increased throughout the 10 second period. This was the expected result as the thermistor was initially left untouched to reach room temperature and then held by a team member during the test. Body heat from their hand warmed the sensor, causing an increase in temperature. From the same figure, the “Distance” column holds data from the lidar sensor starting at 117.6 cm and ending at 30.72 cm, indicating that an object moved towards the sensor, conforming to the expected result as well. A team member held their hand at a distance from the sensor at the beginning of the measurement interval and progressively moved it closer, reducing distance.

On the software side, the necessary backend software was implemented along with a basic mobile application. The Raspberry Pi was connected to AWS IoT Core, allowing for data to be transmitted into the cloud and packaged into a JSON payload, which is then stored in AWS Timestream. The ability to pull data from AWS Timestream into the mobile application was also achieved through the use of the AWS SDK for JavaScript. In our First Deliverable Testing, we demonstrated how the data collected from the sensors was being streamed into an AWS Timstream table and by clicking a button in the mobile application, that data could be pulled and displayed. Separate from the hardware and software integration, authentication functionality was also implemented in the application using AWS Cognito through AWS Amplify. In order to store user data, a GraphQL API connected to a DynamoDB table was created using AWS AppSync through AWS Amplify. A user has the ability to register and login into the application which was shown during the First Deliverable testing by creating an account and having it appear in our AWS Cognito user pool and selecting preferences that were stored in our DynamoDB table.

|  |
| --- |
| Figure 6. Prototype Testing Data on AWS Timestream (6 seconds) |

# Technical Plan

Next semester, our plan will revolve around fully implementing our sensor system, creating our prediction and lidar tripwire algorithm, creating the system enclosure, and completing the mobile application wireframe and implementation.

**Task 1: Microphone**

The last sensor to be added onto our system will be to add our decibel sensor in the form of a USB microphone. This will record noise levels in a room without the need to use GPIO or I2C pins, which are limited on our Raspberry Pi. Once we have the sensor reading data, we will package it into the payload for AWS IoT Core and store it in Timestream. Lead: Ben; Assisting: Allen

**Task 2: Prediction Algorithm**

The data analysis and prediction models will rely heavily on historic data. The model will predict minimum and peak noise, temperature, and headcount values at specific times of day. Lead: Alex; Assisting: Ben

**Task 3: Lidar Tripwire Algorithm**

To create the lidar tripwire, we will need to implement an additional lidar sensor to the Raspberry Pi. This will require us to convert one of the Pi’s GPIO pins into a I2C pin. Once we have two lidar sensors, we will be able to put them in parallel and create an algorithm to analyze if a person is entering or exiting a room. This will be a challenge since there is a lot of room for error when it comes to recognizing a person using just lidar sensors. Lead: Allen; Assisting: Alex

**Task 4: Mobile Application Front-End**

The registration and login portions of the application have been implemented for the most part with the back-end resources deployed and functional, however error validation and account recovery functionality must be added. The rest of the application must still be implemented, with much of the actual application functionality of displaying study locations, user feedback, and other features needing to be designed. Before any more screens are implemented, the main focus will be applied to completing the wireframe on Figma and componsing the user experience. Lead: Ibrahim; Assisting: Allen

**Task 5: System Enclosure**

The system enclosure will house the Raspberry Pi with cutouts for the microphone and thermistor. The enclosure will have cutouts for the lidar wires to be cable managed onto a door frame for headcount estimation. Lead: Ben; Assisting: Ibrahim

**Task 6: Study Space Recommendations**

Users will be recommended study spaces based on personal preference, and will also have the ability to sort study spaces based on specific metrics such as noise level low to high, and vice versa. The sorting can be done by calling specific queries from AWS Timestream in the front end. Lead: Ibrahim; Assisting: Alex

# Budget Estimate

| **Item** | **Description** | **Cost** |
| --- | --- | --- |
| 1 | Raspberry Pi 4 Model B | $55 |
| 2 | Thermistor | $9.95 |
| 3 | 2 x Garmin Lidar-Lite V4 w/ Breakout Board | $150 |
| 4 | Adafruit Mini USB Microphone | $5.95 |
| 5 | Aluminum Heatsink | $1.95 |
| 6 | Electronics Casing | ~$15 |
|  | **Total** | **$237.85** |

The Garmin Lidar-Lite v4 sensor has a unit price of $75. The cost of this sensor is high due to its accuracy and reliability, both of which are reasons we chose it. The Raspberry Pi also has a high component cost of $55. We chose the Pi due to its wide support, documentation, I/O, and processing power.

# Attachments

# Appendix 1 – Engineering Requirements

Team #8 Team Name: Team NoiseHub

Project Name: NoiseHub

| **Requirement** | **Value, range, tolerance, units** |
| --- | --- |
| Case dimensions | 7” L x 5” W x 1.4” H |
| Device Noise | < 30 decibels |
| Data | Predict peak and minimum noise, temperature, and headcount values  Graph 24 hours of historical data |
| Cost | < $500 |
| Accuracy | Headcount accuracy will be greater than 50%  Temperature value +/- 3 degrees celsius of actual value  Noise accuracy +/- 15 decibels |
| Mobile Application | App size will not exceed 500 MB |

# Appendix 2 – Gantt Chart

