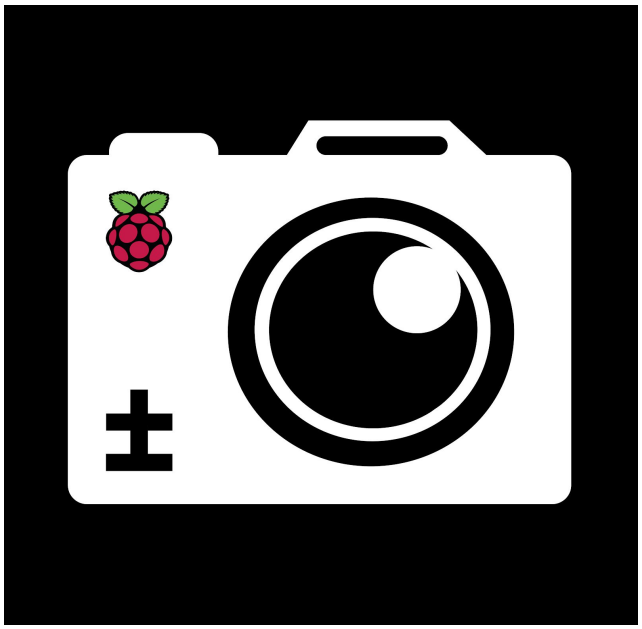


Requirements Specification

Date: 11/6/2020

Team Name: PiWatcher

Project Sponsor: Duane Booher, NAU ITS

Team's Faculty Mentor: Volodymyr Saruta

Team Members: Seth Burchfield, Champ Foronda, Joshua Holguin, Brigham Ray

Version: 1.0

Accepted as baseline requirements for the project:

For the Client:

For the Team:

Table of Contents

1.0 Introduction

Most people are familiar with the idea of “smart” devices, such as Amazon Alexa and Google Home. These are just 2 devices of many that reside in the massive internet of things (IoT) industry. IoT is the idea of networking common appliances or devices that can be made to relay useful information. Data these devices collect is stored in a cloud infrastructure where it can be accessed and used for many desired purposes. The market for these devices is massive and continues to grow unprecedentedly. By 2025, the IoT market is expected to reach over a trillion dollars, up from only 100 billion in 2017.

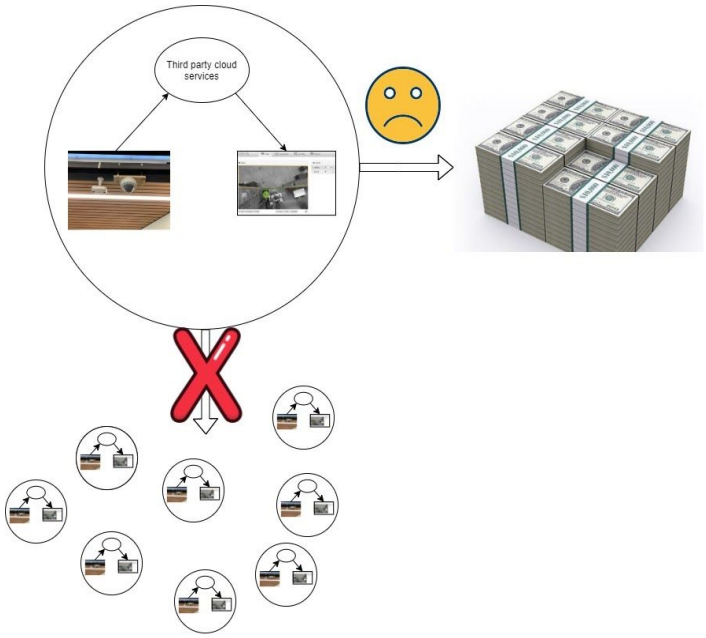
IoT is divided into a few different sectors: industrial, consumer, and business. One might find devices like the Alexa and Home in the consumer sector, while a box transport robot would be found in the industrial sector. The business sector includes devices such as card readers for employee access to buildings, face recognition systems for enhanced security, and clock in / out devices to track employee attendance. PiWatcher is concerned with the business sector of IoT where a people counting infrastructure would be found.

Duane Booher is the IoT team lead at Northern Arizona University. His job is to turn NAU into a smart campus. NAU, like many large organizations, is moving forward with investing in IoT infrastructure to improve workflows campus-wide. One of these investments is NAU’s purchase of a people counting infrastructure for a few of the larger on-campus spaces. This infrastructure resides in places like the Dub, one of NAU’s cafeterias, where large volumes of people flow in and out of the building. The infrastructure can also be found in the skydome, NAU’s indoor football field, for similar reasons. Duane is involved with the devices that represent NAU’s people counting infrastructure since these devices communicate with the network. Right now, the people counting infrastructure currently deployed is a popular commercial solution that provides advanced sensors to count people in buildings, as well as a customizable user interface to analyze and visualize the sensor data. It’s very easy for Duane to customize the commercial interface to suit NAU’s business needs and to provide the university’s residents with data that can keep people safe during the pandemic. The current devices and corresponding user interface are very effective tools that provide NAU with useful information. However, due to the nature of commercial solutions, there are many limitations that prevent organizations like NAU from scaling the people counting infrastructure to more of its rooms and buildings. This prevents Duane from providing useful business metrics to the university. The solution to this scalability problem is the main focus of PiWatcher and is proposed in greater detail throughout the rest of this document.

2.0 Problem Statement

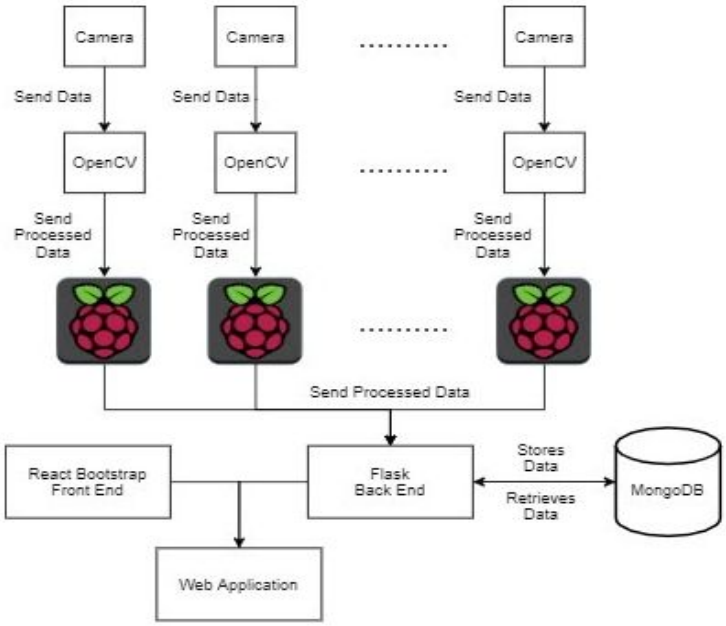
As mentioned previously, the current commercial people counting solution does not adequately suit NAU's business needs. Three issues brought forth by the implementation of the current solution include price, flexibility, and scalability.

- Price:
    - Each individual people counting device is approximately $1000. This makes it very difficult for NAU to implement new devices across campus. There are also likely subscription fees to access the cloud services this solution provides further adding to the cost.
- Flexibility:
    - The current solution offers a great degree of flexibility and customization, which isn't necessarily a problem. However, the user interface this solution provides is closed source and cannot be added to. This prevents NAU from customizing the interface further to suit its business needs.
- Scalability:
    - Cost is the most prohibitive factor when it comes to scaling the current solution. It's not financially viable for NAU to install a high number of the current people counting devices across campus when they can hire a capstone team to produce a more cost-effective solution for free.



3.0 Solution Vision

The proposed solution to these challenges will be a cheaper and scalable people counting infrastructure. This solution will be split into two parts. The first part of the solution will be our people counting devices. Each device will be implemented using a Raspberry Pi paired with a PiCamera. Each of these devices will host a machine learning model that will allow them to accurately count people from a video feed to measure room capacity. The second part of the solution will consist of a user-friendly web application that will be able to store and interpret the live data that we receive. This web application will be able to provide usable business metrics that can be visualized for the users of this application. Down below is an outline of the current envisioned solution.



3.1 People Counting Device

This solution will be able to analyze and count people in a video frame and provide the correct counts of people that have either entered or exited a room. As described in the problem statement, the cost of the current solution is $1000 per device. The solution that we will build will utilize the much cheaper Raspberry Pi alternative.

The Raspberry Pi will be utilizing a machine learning algorithm that can analyze images to detect people in a video frame. As of right now, the current solution is able to count people effectively, so our solution will aim to reach the same level of accuracy to maintain reliability.

3.2 User-friendly Full Stack Web Application

This solution will be able to ingest the information that it receives from each counting device to provide useful metrics for two different groups of users. The first group will consist of public users. These users will have access to seeing the current capacities of certain areas on campus. The second group will consist of internal users at NAU. Internal users will be able to look at visualized information to obtain usable business metrics about the use of areas on campus.

Overall, our solution will consist of:

- A cheap and cost effective people counting device, which will consist of a Raspberry Pi and a PiCamera that will be able to analyze objects (people) from a live video feed. This part of the solution addresses the issues of price and scalability.

- A user-friendly web application, which will serve both the general student population and the business leaders of NAU.

4.0 Domain Requirements

Domain requirements are the highest level, all encompassing elements needed to build the system our client envisions. A short description of each domain requirement is given and will be referenced by the lower level functional, sub-functional, environmental, and non-functional requirements.

D.R.1 User Interface

The user interface of this solution will be clean and easy to use. This interface will provide the ability to make custom adjustments to both the interval of which the Raspberry Pi devices will call to the database and the interval of which the data should be shown to the user from a specific room or building. A user dashboard will keep track of the rooms and buildings that the user is currently taking a look at.

D.R.2 Object Recognition from Camera

Each Raspberry Pi will include software that allows its attached camera to detect people entering the camera's frame of view. This detection is not limited just to one person, but multiple people as well, since in many cases there will be groups of people entering and exiting a specific location. The system will also need to have detection methods employed to avoid irrelevant objects being included in the recorded data.

D.R.3 Count People from Video Frame

The Raspberry Pi will need to be able to store the count of people before it can be sent over the network to the database. Each Pi should store the count in a text file, be able to detect when the text files need to be cleared, and send the local count to the database at the user configured interval. Each entry in the text file will contain information about the count of a room or building at a particular time.

D.R.4 User Authentication

The PiWatcher system will include accounts for authorized users to manipulate the configuration of the system, and to access / remove data that has been created by each of the Raspberry Pi devices. There will be only one account type since the data within the PiWatcher system is sensitive. Unique accounts will be given to each authorized user, and security measures will be put in place to ensure data integrity.

5.0 Functional Requirements

Functional requirements are the more specific requirements corresponding to each different domain requirement. Each functional requirement has sub-functional requirements that go into more detail regarding the makeup of their corresponding functional requirements.

For D.R.1 Elements of the user interface

F.R-1.1 Live data configuration

F.R-1.1.1 - The data is updated often enough that a user will not notice any latency in data collection.

F.R-1.1.2 - By default, the interval that the Raspberry Pi devices will send the people count of a room or building to the database is 5 seconds.

F.R-1.1.3 - An administrator will be able to change this interval depending on their business needs.

F.R-1.1.4 - There will be a few different options to choose from when it comes to the configurable intervals, which will simplify the interface and prevent the user from changing the interval to something that doesn't make sense.

F.R-1.1.5 - The interval is global and will affect the way data is displayed throughout the entire system.

F.R-1.2 Customizable charts

F.R-1.2.1 - Charts can be generated that display the number of people in any given room or building at a time.

F.R-1.2.2 - X axis of these charts will be the user specified interval of time.

F.R-1.2.3 - Y axis will always be the number of people in the location at a specific time.

F.R-1.2.4 - Intervals can be selected from a number of different valid options, but cannot be specified to something that doesn't make sense.

F.R-1.2.5 - Charts can be combined to display data from multiple rooms and buildings.

F.R-1.2.6 - Charts can be added to a user dashboard to save that specific room and building data.

F.R-1.3 User dashboard

F.R-1.3.1 - When a user logs in they will be presented with their dashboard that contains the saved room / building data and custom charts to go along with each of them.

F.R-1.3.2 - The dashboard will contain data from the database that's loaded in directly when the user logs in.

F.R-1.3.3 - There will be a hard limit to the number of custom charts and corresponding data that can be added to the dashboard to prevent performance issues with the interface.

For D.R.2 Elements of Object Recognition

F.R-2.1 Recognize one person

F.R-2.1.1 - The Raspberry Pi will need to be able to recognize if one specific person has entered the camera's frame.

F.R-2.1.2 - As people move in and out of rooms / buildings, they will inevitably enter the frame of view of the cameras.

F.R-2.1.3 - Cameras will be positioned locations near room / building entrances to record people as they enter and exit.

F.R-2.1.4 - Camera needs to detect when a relevant moving object representing a person enters the frame of view.

F.R-2.1.5 - Camera should record the entrance / exit of that object accordingly.

F.R-2.2 Recognize a group

F.R-2.2.1 - The Raspberry Pi will need to be able to recognize if a group of people has entered the camera's frame.

F.R-2.2.2 - Multiple objects will need to be created to represent each person as they enter the frame.

F.R-2.2.3 - The data recorded for group detection needs to represent that a group has entered / exited the specific room / building.

F.R-2.2.4 - Camera needs to be able to detect multiple people entering the camera frame who are going different directions.

F.R-2.2.5 - Camera should record valid data.

F.R-2.3 Avoid irrelevant objects

F.R.2.3.1 - There will be detection methods employed for each camera that prevent irrelevant objects from being recorded as valid data.

F.R.2.3.2 - Camera will be optimized in a way that avoids "detection" of irrelevant objects.

F.R.2.3.3 - Camera focuses on the people entering and exiting the frame of view.

For D.R.3 Elements of People Recording

F.R-3.1 Store count in text file

F.R-3.1.1 - The Pi needs to store an updated people count for a particular location locally.

F.R-3.1.2 - Use of a larger sized SD card is required for each Pi depending on how much data is meant to be kept locally on the Pi.

F.R-3.1.3 - Each text file will have a name which includes the date it was created.

F.R-3.1.4 - New text files will be created on a per day basis.

F.R-3.1.5 - Each count entry in the file will include a timestamp.

F.R-3.1.6 - Each count entry will include the current number of people that corresponds with a timestamp.

F.R-3.1.7 - An entry will also store the building or room so that it mirrors entries you would otherwise find in the database.

F.R-3.2 Storage management

F.R-3.2.1 - A configuration parameter will be provided to determine the amount of local storage should be taken up by the people count text files.

F.R-3.2.2 - When the max count is reached, an algorithm will ensure that what was stored is presently located on the database.

F.R-3.2.3 - The local storage will then be cleared and a new text file will be created in place of what was cleared once the max count is reached.

F.R-3.2.4 - Corresponding network issues will be handled by a systems administrator.

F.R-3.3 Send count to database

F.R-3.3.1 - Once the data has been stored locally, it will be sent to a database.

F.R-3.3.2 - The frequency of which a Pi's people count will be sent to a database is based on the live data configuration parameter mentioned in section F.R-1.1.

F.R-3.3.3 - The Pi will send only that data which has not already been sent.

F.R-3.3.4 - A parameter will be configured to determine where to start looking for brand new entries since it is very likely that the data will be spread out in multiple different files.

F.R-3.3.5 - Before the data is sent, it will need to be checked for errors to ensure data integrity,

F.R-3.3.6 - An administrator will be notified if there are any issues.

For D.R.4 Elements of User Authentication

F.R-4.1 Account management

F.R-4.1.1 - The accounts provided to each individual user will be temporary.

F.R-4.1.2 - The accounts will correspond to the level of clearance a user has at any given time.

F.R-4.1.3 - If someone has access to the PiWatcher system while working at NAU and happens to quit or move to a department where the PiWatcher data is no longer useful for their work, the account is immediately removed from the system by an administrator.

F.R-4.1.4 - There will be no automatic checking for employee transfer of removal so it is up to the administrator of the PiWatcher system to manually manage the accounts.

F.R-4.1.5 - When an account needs to be created, a request will be sent to the administrator to create an account given both a password and user id.

F.R-4.1.6 - No two users can have the same user id, but it is possible for two users to have the same password.

F.R-4.1.7 - A password restriction will be set to ensure that it is very difficult for someone to gain unauthorized access to the PiWatcher system.

F.R-4.1.8 - Server side algorithms will ensure that an account is locked out for a configurable amount of time if a brute force attack is detected.

F.R-4.1.9 - Passwords expire, and once they do, the user whose password expired will need to provide a new, valid password before gaining access to the system.

F.R-4.1.10 - Logins and login attempts will be reported to the database.

F.R-4.2 Account roles

F.R-4.2.1 - The one single role that will be provided for the piwatcher system is administrator.

F.R-4.2.2 - An administrator has unfettered access to the data of the PiWatcher system and the configuration of that system.

F.R-4.2.3 - Only administrators are permitted to manipulate the system.

F.R-4.2.4 - The administrator will also be able to store customized charts and graphs should they choose to.

F.R-4.2.5 - Each account stores customized data on an individual basis.

F.R-4.3 System security

F.R-4.3.1 - Security measures will be employed in addition to account management that prevent the system from being compromised.

F.R-4.3.2 - The REST API can only be accessed with given keys.

F.R-4.3.3 - Raspberry Pis cannot be located on the network

F.R-4.3.4 - Access to the Pi devices other than what is granted to administrators is unavailable

F.R-4.3.5 - All network connections from host to host are secure

F.R-4.3.6 - All security standards are complaint with NAU security

F.R-4.3.7 - Some users can log in without credentials

F.R-4.3.8 - Some users need to log in with their CAS credentials

6.0 Performance (Non-functional) Requirements

Performance requirements detail the specifics of how well a particular aspect of the system will perform. Some elements of the system do not involve aspects of performance, and these are the elements that will not include performance requirements.

For D.R.1 Performance of User Interface

P.R-1 for F.R-1.1 Seamless People Count Updates

P.R-1.1 - It will take less than 10 seconds greater than the update interval for a user to see the most updated data.

P.R-1.2 - Interval updates will take less than 30 seconds to reflect on the user interface.

P.R-1.3 - Interval updates will propagate to all the Raspberry Pis on the system in less than 10 seconds.

P.R-1.3 - Access to a live Pi counter will take less than 50 milliseconds to reflect on the user dashboard.

P.R-1.4 - Download of a Pi's raw data will be equal to the transfer speed of the network connection divided by the size of the Pi's raw data.

P.R-2 for F.R-1.2 Speedy Chart Manipulation

P.R-2.1 - A chart for a room can be generated in less than 100 milliseconds.

P.R-2.2 - A chart for a building can be generated in less than 100 milliseconds times the number of rooms for the building .

P.R-2.2 - An aggregate chart containing both rooms and buildings can be generated in less than 100 milliseconds times the number of rooms the chart contains.

P.R-3 for F.R-1.2 Responsive User Dashboard

P.R-3.1 - The user dashboard will take less than 1 minute to load due to restrictions on the accounts.

P.R-3.2 - Addition of another chart to the dashboard should reflect immediately.

For D.R.2 Performance of Person Recognition

P.R-4 for F.R-2.1 Immediate recognition of an individual

P.R-4.1 - People will be recognized the moment they enter the frame, within 5 milliseconds.

P.R-4.2 - It may be possible for irrelevant objects to be recorded as valid data.

P.R-4.1 - Enter updates will be made to the text file of a particular Pi when a person has crossed an arbitrary line within 10 milliseconds.

P.R-4.1 - Video feed will be continuous and lapse only during system administration.

P.R-4.1 - Exit updates will be made to the text file of a particular Pi when a person has crossed an arbitrary line within 10 milliseconds.

P.R-5 for F.R-2.2 Immediate recognition of groups

P.R-5.1 - Groups of people will be recognized the moment they enter the frame, within 5 times the number of objects recognized, in milliseconds.

P.R-5.2 - It may be possible for clusters of irrelevant objects to be recorded as valid data.

P.R-5.3 - Enter updates will be made to the text file of a particular Pi when a group of people have crossed an arbitrary line within 10 times the number of objects recognized, in milliseconds.

P.R-5.4 - Exit updates will be made to the text file of a particular Pi when a group of people have crossed an arbitrary line within 10 times the number of objects recognized, in milliseconds.

For D.R.3 Performance of people count storage

P.R-6 for F.R-3.1 Local storage performance

P.R-6.1 - Updates to the local text file containing the people count will happen instantaneously.

P.R-7 for F.R-3.2 Performance of local storage management

P.R-7.1 - Deletions of items in Pi's local storage will happen instantaneously.

P.R-8 for F.R-3.3 Performance of database updates

P.R-8.1 - Text files in a Raspberry Pi will be parsed sequentially

P.R-8.2 - Every database update will take less than 10 milliseconds.

P.R-8.3 - The number of database updates will equal the number of lines in a text file times the number of text files.

P.R-8.4 - A bulk update will take the number of lines in each text file that hasn't already been sent to the database times 10 in milliseconds.

7.0 Environmental Requirements

This section covers which platforms the envisioned solution will work on as well as some development constraints.

E.R-1 Usability on Desktop

E.R-1.1 - The user interface will scale appropriately to a variety of desktop environments.

E.R-1.2 - The same interface will be present on the Desktop interface as it is on mobile.

E.R-2 Usability on Mobile

E.R-2.1 - The user interface will scale appropriately to a variety of mobile environments.

E.R-2.2 - The same interface will be present on the mobile interface as it is on desktop.

E.R-3 Host platform

E.R-3.1 - The physical components of the solution will reside on NAU's Flagstaff mountain campus.

E.R-3.2 - The software components of the solution will reside on the NAU network.

8.0 Risk and Feasibility

This section will explore potential risks to the project's success based on the likelihood for the risk to occur and the level of impact it would have on the project. These risks will be evaluated on a scale from low to high. Below will be a table outlining the risks associated with our projects along with a brief explanation of the risks.

Risk	Description	Likelihood	Severity
Accuracy	Accuracy in tracking people in crowds	**High**	**Medium-High**
Reliability of Data	People may “trick” the hardware	**Low**	**Low - Medium**
Overloading Backend	Lag time could be introduced if to many request are sent to the backend at once	**Medium**	**Low**

Risk 1: Accuracy

- *Description:* The accuracy of our model we are going to use can vary based on its training data and parameters it is given. While it may be able to track one or two people at a time, tracking crowds will prove difficult to not only implement but test due to Covid.
- *Likelihood:* There is a high chance that the model will not have 100% accuracy when counting large crowds of people mainly due to our limited ability to test it live and make any adjustments needed.
- *Severity:* This does not show to be a huge problem for our implementation inside of buildings and classrooms. The problem occurs when implemented in a high traffic area such as a lecture hall or somewhere with large amounts of crowded foot traffic.
- *Plan:* Our plan is to put the system in place and test it using as many people as we can to test the accuracy of it. Then based on that feedback go back into the software and make any edits needed to improve the model and raise the

accuracy if needed. We are also looking into testing the model on a sample video feed if possible.

Risk 2: Reliability of Data

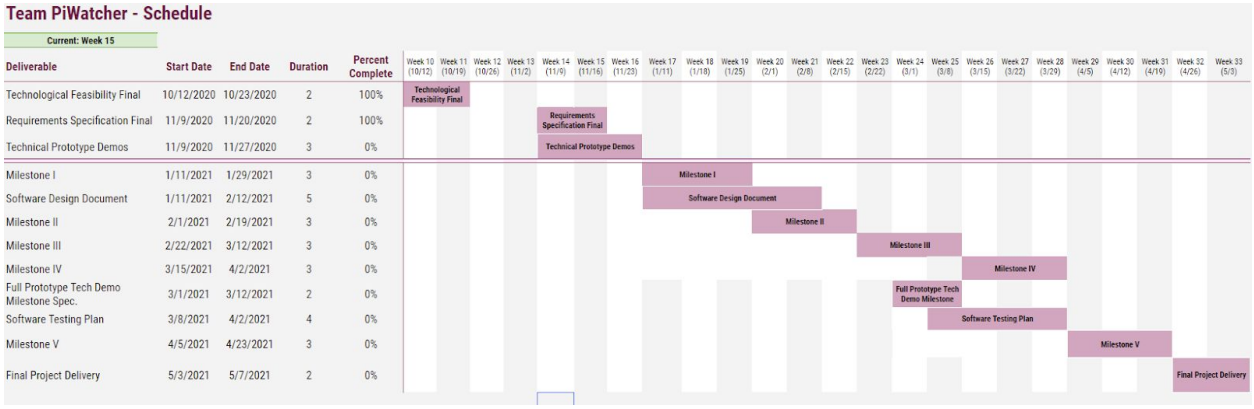
- *Description:* With implementing a new device on campus people may be inclined to try and "trick the software" and hop in and out of frame to give inaccurate counts.
- *Likelihood:* There is a small chance of this happening because we plan to have the system look natural when installed so it cannot be found and exploited easily.
- *Severity:* This is a lower severity as well since the most this will do is give a higher count of people being in a space at a time which can be found out if someone looks at the data.
- *Plan:* We plan to store our data in such a way that will make this case easily identifiable so users will know that the data is not accurate. This includes timestamping when the data was stored in the database.

Risk 3: Overloading Backend

- *Description:* Our backend has to be scalable and thus must be able to handle many Pis hitting it at once. This could cause an increase in lag time if what we are using to host our backend is not reliable.
- *Likelihood:* This has a chance of happening. It mainly depends on what is used to host the backend and how incoming requests are handled.
- *Severity:* We see this of having a low impact for our users. Worst case scenario, users will have to wait a few extra seconds for the backend to get through all the requests.
- *Plan:* We plan to use a software, such as Postman, to send fake data to our backend repeatedly to test and see how it handles the load and at what point it begins to lag, if any.

9.0 Project Plan

This section outlines our current project plan that is necessary for successfully building the envisioned solution. Down below is a Gantt chart that outlines the goals that we have completed, goals we are currently in progress of completing, and goals that we envision including certain milestones for the project.



To ensure that we meet project goals and fulfill the necessary requirements, this project has five defined milestones to ensure that all the goals of the project are met. For each milestone, we aim to implement core features of the application that will be shown to our client during our client meetings. This is to make sure that we are meeting client expectations and to address any bug fixes.

During the mid point of the semester, we hope to have addressed all the requirements needed from the counting device to start deploying them in a test location on campus. From there, we aim to complete the rest of the milestones. Once the milestones are complete, we will aim to fix existing flaws with the product and start performing unit testing and acceptance testing for the product to ensure that it is ready for the final product delivery.

10.0 Conclusion

Many businesses today desire to utilize small, smart IoT devices to record and analyze data relevant to their available resources. The search to maximize the usage to cost ratio of these resources is invaluable, but can not always be achieved in a cost-effective and scalable way with current proprietary options.

Our client, Duane Booher within NAU ITS, has tasked our team with the challenge of creating an open source and cheaper alternative to be incorporated within his goal of making Northern Arizona University a smart campus. Our solution encompasses Raspberry Pi with PiCamera devices and on-device machine learning processing to count and record the amount of people entering and exiting specific areas. A secure, data-centric web application will then provide those metrics to the user in a transparent manner.

This document outlines the requirements that our team has gathered from weekly meetings and months of email communication with our client. All of our broken down requirements have been derived from our domain requirements. These include, instructing our Raspberry Pi devices to recognize objects and count them in a single video frame, as well as creating an accompanying web application that is easy to use and provides secure user authentication. Our team has been working diligently on producing these requirements and we maintain our belief of successfully building an effective people counting solution by our deadline. We are excited to get to produce a platform such as this with the possibility of seeing our devices throughout NAU campus at a future date.

11.0 Glossary of Terms

IoT - The Internet of Things, or IoT, refers to the billions of physical devices around the world that are now connected to the internet, all collecting and sharing data.

NAU - Northern Arizona University.

Open Source - Open-source software is a type of computer software in which source code is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software to anyone and for any purpose.

Cloud Services - Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user.

Machine Learning - Machine learning is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence.

Raspberry Pi - Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation.

Full Stack - Front and back ends of a website or application.

Web Application - A web application (or web app) is application software that runs on a web server, unlike computer-based software programs that are stored locally on the Operating System (OS) of the device.

Scalability - The ability of a computing process to be used or produced in a range of capabilities.

Dashboard - A dashboard is a type of graphical user interface which often provides at-a-glance views of key performance indicators relevant to a particular objective or business process.

Data Integrity - Data integrity is the maintenance of, and the assurance of, the accuracy and consistency of data over its entire life-cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data.

Detection Method - Detection methods represent the variation in a data set in a more manageable form by recognising classes or groups.

Timestamp - A digital record of the time of occurrence of a particular event.

Database - A structured set of data held in a computer, especially one that is accessible in various ways.

Algorithm - A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

Configuration Parameter - Configuration parameters are settings that affect database behavior.

Local Storage - Local storage is the process of storing digital data on physical storage devices, such as hard disc drives (HDDs), solid state drives (SSDs), or external storage devices, such as thumb drives or discs.

Network - A computer network is a group of computers that use a set of common communication protocols over digital interconnections for the purpose of sharing resources located on or provided by the network nodes.

Administrator - A system administrator, or sysadmin, is a person who is responsible for the upkeep, configuration, and reliable operation of computer systems; especially multi-user computers, such as servers.

REST API - Representational state transfer is a software architectural style that defines a set of constraints to be used for creating web services. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the internet.

API Key - An application programming interface key (API key) is a unique identifier used to authenticate a user, developer, or calling program to an API.

CAS - NAU's central authentication system.

Transfer Speed - Transfer speed, the transfer rate is the speed that information is moved between one or more locations. A transfer rate is commonly measured in bits per second (bps) or bytes per second (Bps).

Sequentially - Forming or following in a logical order or sequence.

Desktop - A desktop computer is a personal computer designed for regular use at a single location on or near a desk or table due to its size and power requirements.

Interface - In computing, an interface is a shared boundary across which two or more separate components of a computer system exchange information.

Machine Learning Model - A machine learning model is a file that has been trained to recognize certain types of patterns. The model is trained over a set of data, providing it an algorithm that it can use to reason over and learn from those data.

Parameter - A parameter, generally, is any characteristic that can help in defining or classifying a particular system. That is, a parameter is an element of a system that is useful, or critical, when identifying the system, or when evaluating its performance, status, condition, etc.

Postman - Tool that simplifies each step of the API building process.

Lag - Lag is a slow response from a computer. It can be used to describe any computer that is responding slower than expected.

Gantt Chart - A chart in which a series of horizontal lines shows the amount of work done or production completed in certain periods of time in relation to the amount planned for those periods.

Acceptance Testing - Acceptance Testing is a level of software testing where a system is tested for acceptability.