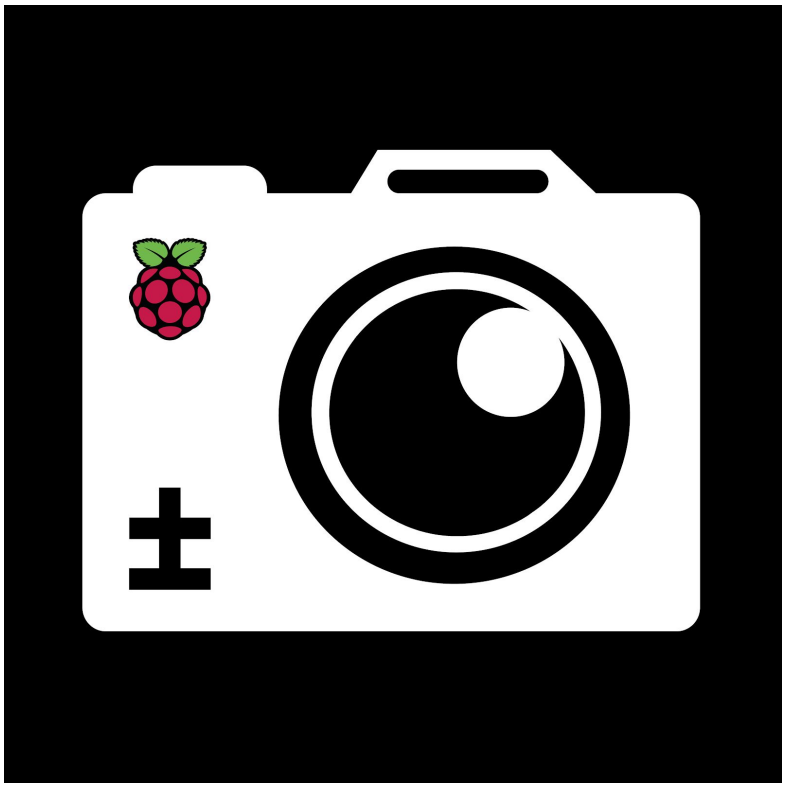

Technological Feasibility:
People Counting Infrastructure

Date: October 23, 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

Table of Contents

1.0 Introduction

The pandemic has made it necessary for large organizations to improve technology that keeps their members safe. One aspect to the pandemic that affects transmission rates is population density. There is currently no straightforward way for large organizations to keep good track of the population flow in and out of their buildings.

Current solutions for controlling population density are incredibly expensive. Each of these solutions utilize a combination of technologies ranging from infrared, lasers, and cameras to keep track of the number of people entering and exiting buildings. This is unfortunate when an organization has multiple buildings because cost limits the number of devices that can be deployed. One organization that is committed to researching better population density tracking technologies is NAU. NAU currently utilizes a technology that costs upwards of $1000 per deployment. This technology provides a people counting infrastructure that allows administrators to keep track of population density, trends, and various other metrics.

Our proposed solution aims to solve the challenges that the current technology does not address. These challenges are price, flexibility, and scalability. Using a Raspberry Pi with an attached PiCamera, each device would cost less than $100 per deployment, allowing the university to scale our people counting infrastructure to nearly all buildings and classrooms. Much like the current solution, our team will provide a web application that would allow administrators to view metrics and trends. The most important aspect of this project is that it will be open source for anyone to use. This allows any organization to utilize the technology in their environments and eliminates their need to purchase expensive, closed source solutions. Our proposed solution may help NAU and large organizations meet their population density tracking needs. The following sections in this document include specific details about each component of the proposed solution.

2.0 Technological Challenges

This section will investigate the design decisions that will be made in order to have a set of technologies that will produce a working product to our client. Below is a brief outline of our technological challenges that will be discussed in this document.

2.1 Object Detection/Tracking Library

The main obstacle our team will face is how we detect and track people moving in and out of spaces. Our system will need to accurately find and track people entering and exiting spaces in real time with a live video feed. This will be done utilizing libraries from languages that give us the fast up to date processing we need.

2.2 IoT Device

An IoT device will need to be chosen to host the object detection library. This IoT device will need to retrieve and process data from a camera that will be attached to it while being inexpensive and easily portable.

2.3 Database Management System (DBMS)

We will need a DBMS to store and access data we gather from tracking people going in and out of spaces. Both our Raspberry Pi and web application will need to have easy access to the database in order to give accurate information to our users.

2.4 Front End Web Application Framework

This project will utilize a front end framework to give a visual representation of the data collected. This framework will need to have a seamless connection with our backend in order to display accurate and up to date data.

2.5 Back End Web Application Framework

The back end framework section will investigate frameworks whose main job is connecting and ingesting data from the database into a usable format for the front end to display. Our solution must be capable of handling an influx of information and managing it properly on demand.

3.0 Technological Analysis

This section will address the major technological issues, challenges, and design decisions that our team has identified as critical to our project's success.

3.1 Object Detection/Tracking Library

The main requirement for our capstone project is to detect and track people from a live video feed. The main point of detecting and tracking people is strictly for counting the occupancy of a room/building thus there is no need for any other type of recognition. The only way to do this is by using multiple Deep Learning algorithms therefore we have decided to use a library previously made in order to get all the information we need. This library will be able to ideally implement both image detection and tracking. If the library is only capable of one, we will need to choose another that can do the other portion.

The main methodology used for detecting objects is a type of Convolutional Neural Network (CNN). This CNN will have two stages: one being extracting certain regions of the video and the second being classifying that region, in our case classifying it as a person. This CNN will work behind the scenes and will be done by the library we decide to use.

3.1.1 Alternatives

The main challenge our team faces with our implementation of object detection and tracking is the following:

- **Ease-of-use**: How easy is the library to use, is it well documented? Are there any tutorials in place already to go off of?
- **Accuracy**: How accurate is the library on a live video feed vice picking people out of a photo?
- **Raspberry Pi Compatibility**: How compatible is this library when implemented on a Raspberry Pi?

3.1.1.1 OpenCV

Open Source Computer Vision Library (OpenCV) is an open source computer vision and machine learning (ML) library. This library offers a variety of algorithms to use for tracking and detecting objects [2]. These libraries can be used as a base to build off of for detecting and tracking people/objects.

Pros

- Vast methods and classes
- Provides set up for tracking people
- Tutorials exist for implementing basic people tracking software on a Raspberry Pi

Cons

- Documentation is not straight forward
- Models not as customizable

3.1.1.2 TensorFlow/TensorFlow Lite

TensorFlow is an open source platform that offers multiple levels of ML models. TensorFlow Lite is a set of tools that assist us in running TensorFlow on IoT devices. This is a more efficient form of TensorFlow specifically optimized for putting and using models on IoT devices [3]. It also offers a variety of pretrained models to choose from so we do not have to create our own.

Pros

- Able to be optimized for a IoT device
- A lot of well done documentation
- Tutorials exist for simple people tracking system

Cons

- Slower compared to competitors
- Can be difficult to debug
- Steep learning curve

3.1.2 Analysis

Our team looked over different tutorials on implementing detection and tracking of objects on a Raspberry Pi. This confirms the Raspberry Pi capability and from the tutorials we see that not much is needed from us to implement the tracking. We mainly need to set up the infrastructure for when the Pi receives a video feed which will require some external libraries but ones that are standard when dealing with ML. As with any ML model there is going to be a learning curve with testing and optimizing our model.

OpenCV's provided documentation is not the friendliest. While it does provide different documentation for different versions the documents themselves are not the easiest to

navigate. They do however provide a lot of examples to go off of for various implementations of their modules [4].
While using TensorFlow, once a model is chosen or created it needs to be run through a converter in order to get it in a TensorFlow Lite format and be compatible with the Raspberry Pi. Then the model can not only be run on our device, but we can also do optimizations to reduce the size and increase the efficiency [3]. This is offered through TensorFlow so nothing will have to be done on our part to optimize our model thus lightening the load and improving the accuracy on the Pi.

Our team ran a quick tutorial to choose, run and optimize a pre-trained model with input data given to us. It then walked us through how to optimize the model with the toolkit provided to ease the process. Most of the work is done by TensorFlow, though we do have the option to make our own model and provide our own data if needed. The API is well documented with descriptions for all methods as well as source code to see what is happening behind the scenes.

3.1.3 Chosen Approach

Technology	Ease-of-Use	Accuracy	Pi Compatibility	Total Score
OpenCV	**4**	3	**5**	**12**
TensorFlow/ TensorFlow Lite	3	**4**	**5**	**12**

Table 1: Object Detection/Tracking Library Summary Table (ratings out of 5)

We decided to choose OpenCV for our library. While the models we create may not be as customizable thus yielding a lower accuracy we believe it will be enough for our use case. Some of our members have used this in the past making the learning curve lower than the other thus making it easier to use in the long run.

3.1.4 Proving Feasibility

For our demo, we will be creating a model to run against a live video feed to begin the tracking of people. We will use predesigned models and data to do this given by other open source resources or within OpenCV itself. After this we will be working on testing it against “crowds” of people to test the accuracy and tweak the model if needed to improve said accuracy.

3.2 IoT Device

In order to retrieve the desired data, we must host our object detection library on an IoT device. The reason we want an IoT device is to limit the amount of hardware needed to perform object detection. With an IoT device it can be done all in one without having to have a separate camera hooked up to a server; we can do everything from the IoT device.

3.2.1 Alternatives

Making the correct decision between these devices rely on the criteria below:

- **Ease-of-use**: How easy is it to install all libraries needed onto the device. Is there an onboard OS made by the company or will we need to install one?

- **Camera Capability**: How easy is it to run a camera on the device? Does it have its own camera module on the board?

- **Network Security**: How easy is it to set up network security? Does it have onboard Wi-Fi/ethernet or will it be separate and need to be configured from scratch?

- **Performance:** How well can the device perform with a ML algorithm running on it? Is there an option to upgrade to a newer version in the future that would yield a better performance?

3.2.1.1 Arduino

Arduinos are an open source electronics platform whose goal is to have easy to use hardware and software. It has its own programming language and IDE built specifically for their boards. They are a bare bones board with high modularity and many modules exist to add onto the board for the project you are doing.

Pros

- Highly modular
- Easy to use
- Onboard software/hardware
- Inexpensive
- Cross platform (OS)

Cons

- Needs other modules to function how we want
- Does not have an easy way to attach a camera
- Does not come with a lot of options out of the box

3.2.1.2 Raspberry Pi

Raspberry Pi(s) are essentially a mini computer. Their boards vary from complex to minimalistic. Some standout features include onboard ethernet and USB 3.0 compatibility. They can be used as a mini computer or configured to be a mini server as well. They are very versatile with a lot of potential when it comes to projects.

Pros

- Highly Modular
- Has its own OS
- Easy to use
- Inexpensive
- Can be used as a server
- Onboard ethernet
- Has OEM camera that attaches directly to the board

Cons

- No mount for on board camera
- More ram and processing power will cost more money

3.2.1.3 Coral

Coral is a board with a focus on AI applications and hardware that is meant to deal with it. This device is mainly meant for prototyping while its modules are used to be added onto other boards. These modules assist in the deployment of an ML model. This product is a mini computer with a lot of power and capability for its size.

Pros

- Is meant to deal with machine learning
- Easy to use Linux like environment
- A lot of capability when it comes to object detection

Cons

- Not inexpensive ($129) [15]
- Onboard camera is low quality and will need to look at other options
- Add-ons Coral offers are more cost efficient
- Add-ons can be utilized on other IoT devices

3.2.2 Analysis

Arduinos offer a lot of scalability, however, this comes at a cost of having to add different modules for different purposes. Thus, for our use case we will need to purchase different modules not necessarily offered by the Arduino itself, for example, a camera. This is not ideal since we will have to find a camera that is not invasive and compatible with the board. The onboard software will most likely not be utilized as well

since we need a language that can do ML models which the onboard language cannot. From the research our team has gathered, network security on these boards are not the most intuitive and have proven to be a challenge.

During our research into Raspberry Pi(s) we found the company makes an onboard camera worthy of use for our use case and can easily be installed and accessed with a few lines of code. The Pi(s) also come with their own OS and can be run using versions of Linux and Windows. They also have their own HDMI output so the Pi can be easily displayed to a monitor. Both our team and client have experience with Pi(s), lowering the overall learning curve. Our client also has resources to configure the Pi to the network we will be using thus eliminating network security as a potential pain point. Performance on the Pi may dip when running the model however if needed an external module could be attached and used to handle the workload. Its capability to also act as a server for other purposes in our project is an added bonus.

Coral's prototyping board is mainly used for machine learning, but lacks in all other prospects. Its onboard camera is too low a resolution to use, thus creating a pain point in finding a new one. It is also limited to running on Linux as well as needing a Linux machine to access the device in the first place. What is promising is the external comments Coral offers. They have many modules outside of their boards that are used to assist in the ML process.

3.2.3 Chosen Approach

Technology	Ease-of-Use	Camera Capability	Network Security	Performance	Total Score
Arduino	3	3	3	4	13
Raspberry Pi	**4**	**5**	**5**	4	**22**
Coral	2	2	2	**5**	11

Table 2: IoT Device Summary Table (ratings out of 5)

Our team chose to pick the Raspberry Pi as our IoT device mainly due to the onboard camera being what we need and the ease of installation compared to the other boards. The network security portion is also already taken care of for us as well as our team having prior experience with this type of device. We believe that this device will provide all that we need for processing and relaying the data we need and is flexible enough to be used in other ways.

3.2.4 Proving Feasibility

For our demo we plan to host the object detection library that we choose on the Raspberry Pi and have it relay the data to a database locally hosted if one is not set up on a server at the time of the demonstration. After the demo, we will work on optimizing the Pi to run more efficiently and look to see if external devices are needed to handle the ML workload. We will also look at the possibility of hosting our front and back end on a Pi configured to be a server.

3.3 Database Management System (DBMS)

For our platform to track the number of people passing through our cameras, an efficient database management system with quick query times is essential. The sometimes massive data throughput from our many Raspberry Pi devices must not bottleneck our platform or miscounts will be probable and the accuracy of the platform as a whole is tarnished. As well as speed, the DBMS needs to be scalable and have the ability to easily register and remove endpoint data. The future scope of this project could include hundreds to thousands of endpoints and the database should be able to handle the registration successfully.

Connected through our backend, the DBMS will record the given data into a database and update the entries on a regular basis. When connected to our front end, the databases will provide the necessary data to give the user the ability to check the counts often and in real time. For our team to achieve an optimal system, the DBMS needs to be correct for the job. *MongoDB, CouchDB, MySQL*, and *PostgreSQL* are the top four of which our team has found during our research for a suitable DBMS.

3.3.1 Alternatives

The main challenges our team faces with our implementation of a DBMS are the following:

- **Ease-of-Use:** Is the database easy to implement? Are endpoints easy to add and remove from the system?
- **Compatibility:** Is the DBMS compatible with our chosen language and schema?
- **Performance:** Does the database system have acceptable performance for our situation?

- **Cost:** What does the DBMS cost? Is it open source?

3.3.1.1 MongoDB

MongoDB is a popular general purpose NoSQL database focused on scalability and ease of development. This DBMS was conceived in 2007 by the team at DoubleClick to suit their needs of a less rigid schema. MongoDB is now developed by MongoDB Inc. and is open source. The database system is written in C++. [9]

Pros

- Extensive documentation
- Large community
- Multiple language support
- Open source (free to use)
- Fast
- Scalable
- Flexible document model

Cons

- Does not have expansive data replication features

3.3.1.2 CouchDB

CouchDB is an open source NoSQL DBMS by Apache. Their tagline is “relax” to emphasize their focus on reliability and data safety [11]. It was originally introduced as an Apache Software Foundation project in 2008 [10]. The database is written in Erlang and is open source.

Pros

- Extensive documentation
- Multiple language support
- Open source (free to use)
- Better data safety features
- Flexible document model
- Large community

Cons

- Not as a fast
- Less scalable

3.3.1.3 MySQL

MySQL is possibly the most used database system of all. Originally released in 1995 by the Swedish company, MySQL AB, it has since been acquired by Oracle and is one of the go-to RDBMS systems with its inclusion in the LAMP (Linux, Apache, MySQL,

PHP) stack. Written in C and C++, the RDBMS is open source and free to use. [12]

Pros

- Extensive documentation
- Easy to use
- Large community due to age
- Multiple language support
- Open source (free to use)

Cons

- Only relational model could be too rigid for our application

3.3.1.4 PostgreSQL

PostgreSQL is an ORDBMS, object relational database management system, with more than 30 years of development time behind it. Created in the late 1980's at UC Berkeley, the database system "is considered the most advanced database engine today." [13] It supports both relational and non-relational data structures and is one of the four technologies within the PERN (PostgreSQL, Express, React, Node.JS) stack. This database system is written in C and open source for public use.

Pros

- Extensive documentation
- Supports both relational and non-relational structures
- Large community due to age
- Open source (free to use)

Cons

- Read times can be slower than other options
- Less language support

3.3.2 Analysis

From our research, we have found that each of our prospective databases is representative of a specific niche within the DBMS spectrum. All four have shared characteristics, such as being open source and being highly compatible with different languages and libraries. Yet, their differences are what separate them from qualifying better or worse for our project.

MongoDB and CouchDB are both document model databases with flexible (or no) schema with large amounts of documentation available, yet they market themselves as having different purposes. MongoDB is a DBMS created for performance with easy project scaling and quick query times. CouchDB is slightly less about those characteristics and more about data safety and reliability. Our third option, MySQL is a relational model database with a rigid schema, but with decades of backing and work

behind it. This database software seems to be the go-to for many developers looking for a quick and simple database to employ in their project. Our final database system is PostgreSQL, another with decades of development time and a piece of the PERN stack. The database system is object relational, thus providing the option for both relational and document-based storage. This feature could potentially be a huge benefit for us in our project if we choose to use PostgreSQL.

3.3.3 Chosen Approach

Technology	Ease-of-Use	Compatibility	Performance	Cost	Score
MongoDB	4	**5**	**4**	**5**	**18**
CouchDB	4	4	3	**5**	16
MySQL	4	4	3	**5**	16
PostgreSQL	**5**	3	**4**	**5**	17

Table 3: Database Management System (DBMS) Summary Table (ratings out of 5)

MongoDB will be the chosen database management system for our project due to its document-based storage platform. The DBMS is a fast, scalable option with lots of documentation and a large community behind it. A close second is PostgreSQL with its object relational system and will be our next choice should MongoDB end up unacceptable.

3.3.4 Proving Feasibility

To prove the feasibility of our chosen DBMS, we will first check for the successful connection with our front end and back end to be sure they are compatible. We will then check the database's performance by attempting to simulate the data collection of the Raspberry Pi endpoints. By sending queries at various intervals, we can get an idea about how the database will respond in a real-world scenario. In the event of MongoDB not satisfying our requirements, PostgreSQL will be given a second look for testing.

3.4 Front End Web Application Framework

Providing the users of our people counting solution with a professional, clean looking user interface is paramount to its effectiveness. Luckily, there is no shortage of libraries and frameworks to aid in the process of user interface design and creation. A good user

interface design is simple to understand, useful for the customer's needs, and should be able to provide them with accurate information in a well thought out format. The interface should be responsive and consistently refresh to collect the most up to date information from the back end of the people counting solution. The front end should also be mobile friendly, although the focus of the front end design for this particular project will be toward desktop users. We'll most likely be refining the design of this front end to reflect the ideas of our client, but make sure to focus specifically on the base functionality and user experience before more advanced features are added.

3.4.1 Alternatives

Upon further investigation, there seem to be a few different options available when it comes to front end application frameworks. Each technology listed has its pros and cons, but there is a consensus on what requirements we would like to abide by when utilizing a particular front end library. These libraries are: *Bootstrap*, *React*, and *Angular*. The requirements that we would like to fulfill when it comes to utilizing these technologies are:

- **Performance:** How responsive is the front end library? When a user executes an event on the sight, are they able to quickly access the data that they are requesting?

- **Ease-of-Use**: How quickly can the customer access the data that they need? Is the data presented cleanly such that it is not a headache to interact with the system? Does the data cleanly represent what the database contains at any given time the user decides to log into the web interface?

- **Data integrity**: Does the front end contain accurate information from the database? Are there tools available for representing the data? Is the user allowed to access all the data or are there restrictions? What kind of data should be represented?

- **Learning Curve**: Is the front end simple enough to design so that we won't get bogged down trying to understand complicated language libraries, and be able to present an adequate solution to the client within a given time frame?

3.4.1.1 Bootstrap

Bootstrap is a front end tool kit that simplifies nearly everything about the layout and design portions of a website. The library focuses exclusively on making the construction

of the website as simple as knowing the classes provided by Bootstrap. This allows the developer to focus less on the technical aspects of CSS and JavaScript since Bootstrap is primarily composed of these two front end languages [16].

A website becomes increasingly challenging to develop when mobile devices are considered. Bootstrap takes complete care of this issue by providing classes that are both desktop and mobile responsive. Even though the focus of this solution deals with a desktop front end, having the front end scale down to mobile devices is a nice bonus. There is no need to dig into complicated CSS media queries, Bootstrap takes care of that for you. In fact, the developer does not really need to use CSS at all for design unless you care to implement something specific.

It can be cumbersome to implement a Bootstrap site that allows for excellent performance. Other event driven JavaScript intensive libraries are most likely going to be more effective when performance is the driving force behind the design of a particular site. It is possible to merge both a front end library with an event driven library, such as React JS. This could end up happening for this particular people counting project.

Although the more developed templates that are built using the bootstrap framework may cost money, Bootstrap is mainly a free tool to use and develop with.

Pros

- Easy to use
- Desktop and mobile compatible
- Extensible
- Very responsive
- Extensive documentation
- Free

Cons

- Not the best performance
- Void of back end communication library
- Not meant for enterprise level development

3.4.1.2 React

React is Facebook's way of dealing with interactive websites. There are many advantages to utilizing this powerful framework to build responsive websites that include a lot of user interaction. It may be useful for us to implement a front end solution that includes something like React JS [17].

The efficiency of React is one of its biggest pluses. When a user interacts with a site built on React, only the component that the user interacted with at that point is updated. This prevents the entire page from having to reload when the user interacts with

anything on the website. React can improve performance dramatically, especially for something with less processing capability such as a mobile device. These components can be stateful, such that, when a user interacts with them, the state of said components are updated. Once a component is developed, it can be used throughout the site wherever it is necessary. This makes websites scalable, and because of the nature of components in general, more complex components can be built upon simple foundations.

There are, however, a few cons to consider when dealing with React JS. One of these is a steep learning curve that includes how to structure React components. The style of language used to structure components is a bit complex, especially for someone just getting into programming or learning how to write in JavaScript for the first time. In fact, a new way of styling JS code needs to be learned, called JSX. The braces and code that reside within this particular styling are a bit different than what would be expected from vanilla JavaScript styling.

React is not the only front end JavaScript library available on the market. Because of this, it is possible for alternatives to come up that may have less of a learning curve or better functionality for our customer's needs.

Pros

- Backed by Facebook
- Reusable components
- Scalable
- Multi-page
- Performance Oriented
- Extensive documentation
- Free

Cons

- Steep learning curve
- Not automatically mobile compatible

3.4.1.3 Angular

Angular is a single page web application framework that was developed by Google. Using a tool like Angular ensures that you'll be able to accomplish what you want with the backing of the massive amount of development Google has poured into the platform. It includes many features such as tooling for communicating with the back end as well as a powerful command line that allows you to develop quickly and efficiently. This can also be a con, however. Given the expansive tooling around angular, it is easy to find multiple different ways to do one specific task. Angular's documentation is thorough, but it can be irritating to start building a feature and realize that you have been implementing it with two different methods at once [18].

One important feature of Angular is that it is easy to scale web applications. Tools are included that allow for development on both desktop and mobile platforms. The structure of Angular is such that the client is able to load what they need locally with minimal communication with the server. This makes Angular applications very snappy, which from a client's point of view, is a very good thing!

One problem with Angular is that it is focused around single page web applications. While this is good for many projects, ours will be built around the idea of multiple user views, so Angular in our case is not a great pick for a front end framework. Another issue with Angular is the steep learning curve associated with its initial implementation. This comes with most front end frameworks, but Angular is especially difficult to learn. In our case, we would like to minimize the impact of spending time learning difficult technologies, yet another reason why Angular is not ideal for our purposes.

Pros

- Backed by Google
- Reusable components
- Scalable
- Performance oriented
- Extensive documentation
- Free

Cons

- Steep learning curve
- Single page app oriented
- Documentation redundancy
- Divided community

3.4.2 Analysis

The three front end frameworks mentioned in this document are all fully supported tools that simplify the front end development process significantly over raw full stack development. Each different framework can be used in different ways, for different purposes. It is clear after going through each of these technologies that there are a couple options that speak to us more clearly than the other.

Learning curve is one common aspect of these tools that our team is focused on. Only one of these tools, Bootstrap, has a very slight learning curve. In fact, it is so easy to learn bootstrap, it is likely easier to build a bootstrap web page than it is to use raw HTML, CSS, and JavaScript. Both Angular and React are difficult tools to learn. That is not to say Angular and React are not useful or powerful, they certainly are, but unfortunately that means we'll have to focus on the one whose learning curve is the slightest.

Front end frameworks can often overlap, and that is what we see with Bootstrap and React. There is an entire development combining the two frameworks called React Bootstrap that takes the advantages of both frameworks and pools them into one more powerful framework. It is likely that we will end up using this tool to simplify the development process.

3.3.3 Chosen Approach

Technology	Performance	Ease-of-Use	Data Integrity	Learning Curve	Total Score
Bootstrap	3	**5**	**5**	**5**	**18**
React	**5**	4	**5**	4	**18**
Angular	5	2	5	1	13

Table 4: Front End Framework Summary Table (ratings out of 5)

Strangely enough, there is no one chosen approach between the three frameworks. Both React and Bootstrap have their place for different aspects of web development. While React is focused around the performance and responsiveness of websites, Bootstrap is focused on layout and design for both mobile and desktop devices. Bootstrap is quite easy to learn while React has a decently sized learning curve. Luckily, there is a development in the works called React Bootstrap that combines the two frameworks in such a way that allows for both responsiveness and elegant design.

3.4.4 Proving Feasibility

In order to prove the feasibility of React and Bootstrap, we will be learning how to develop in these languages through documentation analysis and building a sample website. This will allow us to judge more adequately what we can expect in terms of a learning curve for getting into development using these languages. Since they are free, we know right off that bat that we do not need to get into contact with our client to help us fund these tools which is a big deal. Web debuggers on Firefox and Chrome offer metrics on performance so we will be able to determine how responsive the sample site is as we develop it. To make sure that we can make server-side calls, we will implement code that tests this feature. All in all, it should be fairly easy to determine the feasibility of these two web technologies.

3.5 Back End Web Application Framework

The ecosystem of our project solution is going to be composed of Raspberry Pi(s), a web application, and a database. Our product needs to have a back end framework to connect all these pieces of the overall product together. Each of these Raspberry Pi(s) will need to communicate and send processed video feed data to the database and web application. Since our solution is going to utilize multiple Raspberry Pi(s) across multiple buildings on campus and potentially the Flagstaff community, we need to make sure that these devices are able to communicate effectively without bottlenecks.

3.5.1 Alternatives

During our research, the potential backend frameworks that were found that could address our technological issues and challenges are: *Django*, *Flask*, and *Express*.

Making the correct decision between these frameworks rely on key criteria. The identified key criteria are:

- **Ease-of-use**: Is the back end framework easy to use? What is the learning curve of this framework? How large is the documentation? Does it have a well established community?

- **Scalability**: How does it handle multiple connections coming in? Will it be able to handle the load?

- **Security**: How can we utilize the framework so that the interaction between the users and the web application is secured?

- **Performance:** How fast is the framework? Does the performance decrease when the application scales?

3.5.1.1 Django

Django is a popular and widely used free and open source web application framework that utilizes the Python programming language. This web framework was created back in 2005 to allow developers to build web applications rapidly with an emphasis on security and scalability [5]. Along with that, Django also provides plugins, modules, and

a front end framework that follows a model-template-view architectural pattern.

Pros

- Scalable
- Focuses on security to prevent things like (SQL injections and XSS)
- Rich documentation
- Large community
- Provides user authentication

Cons

- Steep learning curve
- Performance dependent on good design
- Not good for static one-pagers and microservices [6]

3.5.1.2 Flask

Flask is another popular free open source Python-based web application microframework that was initially released in 2010 to develop simple and small web applications. Flask keeps the core of the back end simple while also making it extensible allowing us to add our own functionality and libraries to allow our application to work effectively. This allows us to easily add and integrate numerous extensions that Flask does not provide such as database integration, form validation, uploading handling, open authentication technologies and more [7]. Flask provides useful documentation on how to install, setup, and get started on a simple application. They also provide documentation on how to test Flask applications.

Pros

- Minimalistic and lightweight
- Easy to learn and easy to set up
- Lots of useful documentation
- Very flexible and extensible

Cons

- Does not provide much out of the package
- Need to find extensions for database integration, form validation, authentication, etc.
- Not-asynchronous - can only handle a single request at a time

3.5.1.3 Express

Express is a popular free open source web application framework that utilizes Node.js that was initially released in 2010. Express paired with Node.js allows developers to easily create robust API quickly using JavaScript. Along with that, Express has extensive documentation on how to install the framework, how to utilize the tools it provides, guides on how to create routes, and so much more [8]. On top of the easy learning curve and great documentation, Express also allows support for multiple

plugins while also being lightweight and flexible.

Pros

- Minimalistic and lightweight
- Easy to learn and easy to set up
- Lots of useful documentation
- Very flexible and extensible
- Large community

Cons

- Does not provide a lot as it has very few dependencies

3.5.2 Analysis

After further analysis into each tech, we can easily see that these three frameworks are popular, supplied with rich active communities, and are able to solve the technological challenges that our product faces.

Comparing between Express and Flask, we can easily see that they provide the same pros and cons with the only difference being the programming language that is used. Both Express and Flask offer a lightweight and minimalistic option that is easy to learn and easy to set up. Since the framework is very lightweight and minimalistic, the framework is a lot easier to learn than other frameworks. However, further analysis on different libraries/frameworks to use for database integration, user authentication, and many more needs to be inspected since they do not already provide these tools.

Taking that into consideration, Django offers a scalable and secure option. Just like Flask, Django utilizes Python for development. In contrast to Express and Flask, Django comes preloaded with packages for database integration, user authentication, and its own front end framework. However, because of that Django has a steep learning curve in comparison to Express and Flask.

In terms of scalability, large companies such as Pinterest, Instagram, Netflix, and IBM utilize these technologies. However, Django is often recommended over Flask if you are looking to build an application that needs to scale. However, the scalability of Django is completely dependent on the size of the architecture. If the architecture is designed badly, then performance and scalability suffer. For the purpose of the solution, we need to utilize the backend to create a REST endpoint. This means that the packages that Django provides are unnecessary. Along with that, since Django provides a front end library built-in, it will not be compatible with the front end library that we choose in Table 4.

3.5.3 Chosen Approach

Technology	Ease-of-Use	Scalability	Security	Performance	Total Score
Django	3	**4**	**5**	4	16
Flask	**5**	**4**	3	**5**	**17**
Express	4	**4**	3	**5**	16

Table 5: Back End Framework Summary Table (ratings out of 5)

The chosen approach that we will be using for the back end framework for our product is Flask. As discussed in our analysis, Flask offers the correct amount of tools that we need, while also not sacrificing a lot of performance.

3.5.4 Proving Feasibility

In order to show that Flask is the best option for our solution, our teams will create a demo website. We will use Flask to connect our demo website and our chosen database together along with a test REST API to provide a place for stress testing the back end of our application. This stress test will check the application performance and simulate the amount of connections that our back end will need to handle from the Raspberry Pi(s). In the scenario where Flask does not provide the metrics that we need, we will use the same approach for the second alternative, Django.

4.0 Technological Integration

This section defines how our team will address the following integration challenges on how our platform's pieces will fit together as well as what challenges we aim to solve with each of them.

4.1 Integration Challenges

All of our decisions regarding the design of our platform will have to run through one final challenge: "will our software choices run together efficiently and effectively?". To ensure that these software choices are able to run together, our team has outlined the envisioned product to address our integration challenge.

4.2 Envisioned Product

Our platform as a whole will begin at the endpoints with multiple Raspberry Pi devices and PiCameras utilizing the OpenCV library. These endpoints will process how many people enter and exit an enclosed space. The processed data will be sent to our Flask back end. Flask will be our link between the front end web service and our database management system. Once the processed data has been received, MongoDB will be sent the data to be recorded into the database. Every update from each endpoint will immediately be sent to our Flask back end and queried into our database. From there, we will utilize our React based front end tied with our Flask back end to interpret the data such that it can be displayed in an appealing and informational manner to the user.

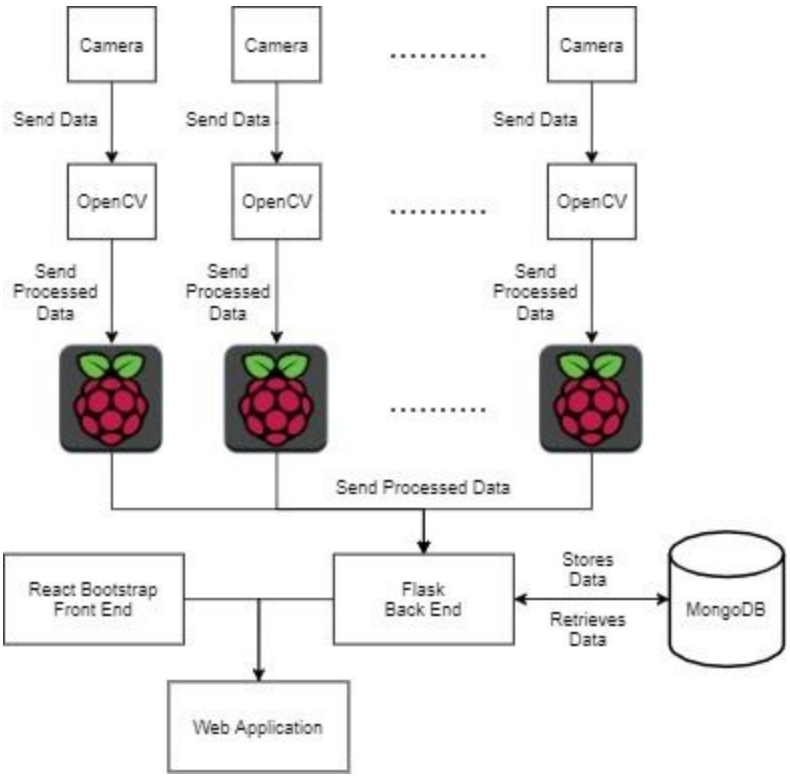

Figure 1: Diagram of envisioned product. This diagram outlines the workflow of our envisioned solution. The camera sends data into OpenCV which sends the processed data to our Raspberry Pi. All these devices will send processed data to the back end of our web application which will be used to display the appropriate data analytics.

5.0 Conclusion

Our project's purpose is to provide an effective solution to monitoring room population count and space utilization during the COVID era and beyond. Our goal with this document is to gain more knowledge of the many technologies available to be incorporated into our platform and choose our prospective solution. Our solution consists of React-based front end, Flask-based back end web services, MongoDB as our database management system, and OpenCV as our object detection library. These software frameworks and libraries will be used with our chosen IoT device, Raspberry Pi, to analyze images on a camera and provide the necessary information for our solution to function. With the software we have chosen above, our team is confident our solution will successfully solve our problem. PiWatcher will be a cheap, effective, and accurate people counting software platform.

6.0 References

[1] OpenCV People Counter. (2018). Retrieved October 1, 2020, from https://www.pyimagesearch.com/2018/08/13/opencv-people-counter/

[2] About. (2020). Retrieved October 1, 2020, from https://opencv.org/about/

[3] TensorFlow Lite guide. (2020). Retrieved October 5, 2020 from https://www.tensorflow.org/lite/guide

[4] OpenCV Modules. (2020). Retrieved October 6, 2020 from https://docs.opencv.org/master/index.html

[5] Django Project. (2020). Retrieved October 7, 2020 from https://djangoproject.com

[6] Django Pros and Cons. (2020). Retrieved October 7, 2020 from https://www.netguru.com/blog/django-pros-and-cons

[7] Flask Background Information. (2020). Retrieved October 7, 2020 from https://flask.palletsprojects.com/en/1.1.x/foreword/

[8] ExpressJS. (2020). Retrieved October 7, 2020 from https://expressjs.com/

[9] MongoDB. (2020). Retrieved October 7, 2020 from https://docs.mongodb.com/manual/introduction/

[10] IBM Cloud Learn Hub. (2020). Retrieved October 8, 2020 from https://www.ibm.com/cloud/learn/couchdb

[11] CouchDB. (2020). Retrieved October 7, 2020 from https://couchdb.apache.org/

[12] What is MySQL?. (2020). Retrieved October 8, 2020 from https://www.atlantic.net/dedicated-server-hosting/what-is-mysql/

[13] PostgreSQL vs MySQL. (2020). Retrieved October 13, 2020 from https://www.xplenty.com/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/

[14] What is Arduino? (2020). Retrieved October 15, 2020 from https://www.arduino.cc/en/Guide/Introduction

[15] Products. (2020). Retrieved October 12, 2020 from https://coral.ai/products/

[16] Build fast, responsive sites with Bootstrap. (2020). Retrieved October 12, 2020 from https://getbootstrap.com

[17] React A JavaScript library for building user interfaces. (2020). Retrieved October 12, 2020 from https://reactjs.org/

[18] Angular Features & Benefits. (2020). Retrieved October 12, 2020 from https://angular.io/features