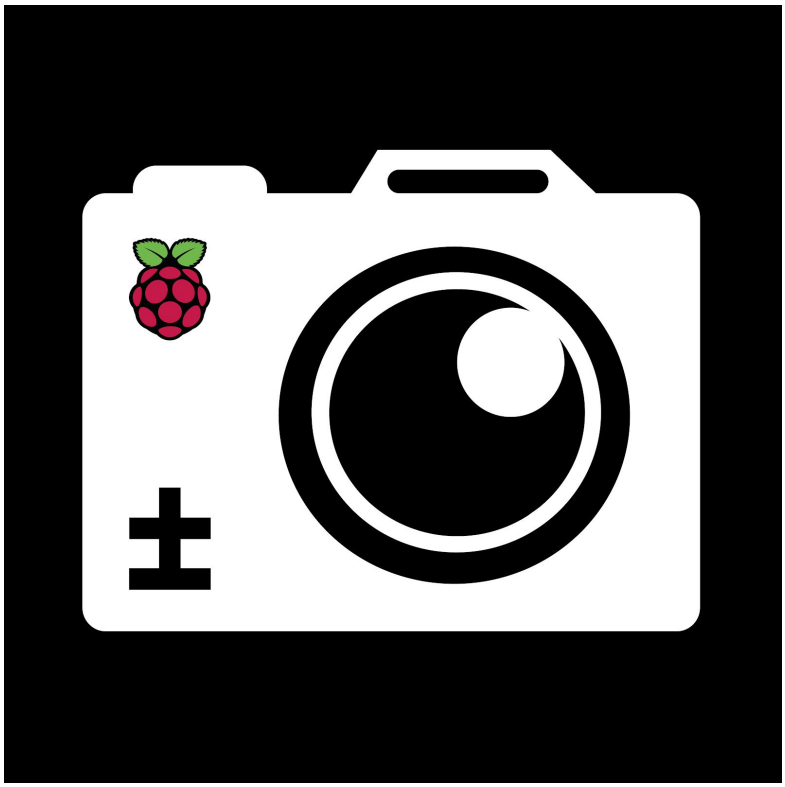

Software Design Document
People Counting Infrastructure
Version 1.1

Date: February 12th, 2021

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, Brandon Thomas

Table of Contents

1.0 Introduction

These trying times have deemed it necessary for large organizations to provide technologies that will help keep people safe. Currently, there isn't any way for large organizations to keep track of population flow without enormous investment. This is where our PiWatcher system steps in to fill the gap.

Current solutions for controlling population density are expensive. They utilize a number of complex technologies to help count people, some of which are unnecessary and further add to the cost. This is unfortunate when an organization has multiple buildings because this cost limits the number of devices that can be deployed, and therefore the number of locations that NAU can keep safe.

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 around $100 per deployment, allowing the university to scale our people counting infrastructure to nearly all buildings and classrooms.

For this solution to be effective, a web application will be provided for administrators to customize the flow of information from the Raspberry Pis to suit their organization's business needs. These customizations will range from the ability to change the rate at which the Pis count people to the types of graphs that will display once a user decides to view the metrics provided by one of the Pis. Some data will be stored in the user's browser so that they are able to view data from certain rooms and buildings that they were looking at during a previous time. Of course, this particular interface will need to be responsive and utilize the latest web technologies to keep the site secure.

All the data provided in the web application will be processed using flask so that the Pis will be able to asynchronously send data. Different endpoints on the flask backend will streamline the amount of data being sent to the user so that their web application remains responsive. MongoDB will store this sent data in JSON format which is easily translated to human readable information once it is sent to the web application. These technologies are capable of handling a high volume of POST requests which may be necessary once a large number of IoT devices are deployed on campus. Only users that obtain an API key will be able to access this backend.

Most importantly there remains the Raspberry Pi portion of the solution. Each Raspberry Pi will be configured to collect data at a specific location. Data will be recorded using a camera and processed using machine learning technologies. Image recognition technologies will determine the number of people in a frame and store that

data locally for a short period of time. Afterwards, it will be sent over the NAU network in JSON format to be stored in the MongoDB database.

The components of this solution will reside on the NAU Mountain campus. Ideally the Raspberry Pis will maintain a low profile as to not alarm the students or faculty. The application that administrators utilize to perform data analytics that this solution provides will scale appropriately to both desktop and mobile devices.

2.0 Implementation Overview

Our proposed solution includes utilizing an IoT device and camera to accurately count people within buildings and classrooms, utilizing OpenCV, an open source computer vision library, and TensorFlowLite, an open source deep learning framework for IoT devices. In combination these libraries and hardware will be placed in buildings and classrooms strategically in order to get the most accurate count possible.

The information gathered from these devices will be sent to a Flask backend that is tied with a MongoDB database. Thus, all information is through our backend and not the IoT device itself. This backend will then clean up the data as necessary to prepare it for ingestion through the front end.

Our front end will be utilizing React to create an elegant login and landing page with search functionality. The login will be utilizing our own authentication system so only authorized persons will be able to access it. Then there will be a search functionality to get specific buildings and classrooms within said building. This information will then be displayed via a graphical representation along with a fraction and percentage of the current capacity. The graph will also be configurable to the user so they can give specific time intervals and the data will then be displayed as needed.

This is all being done to help our client build and utilize a system for keeping record of how spaces within buildings are used for both the current pandemic and to find a way to best utilize these spaces in the future based on data gathered. For example we can allocate more people to certain spaces for club meetings or create space for offices for certain departments to be put depending on the space at hand. Through the use of IoT devices this offers an inexpensive but resourceful way to accomplish this task.

3.0 Architectural Overview

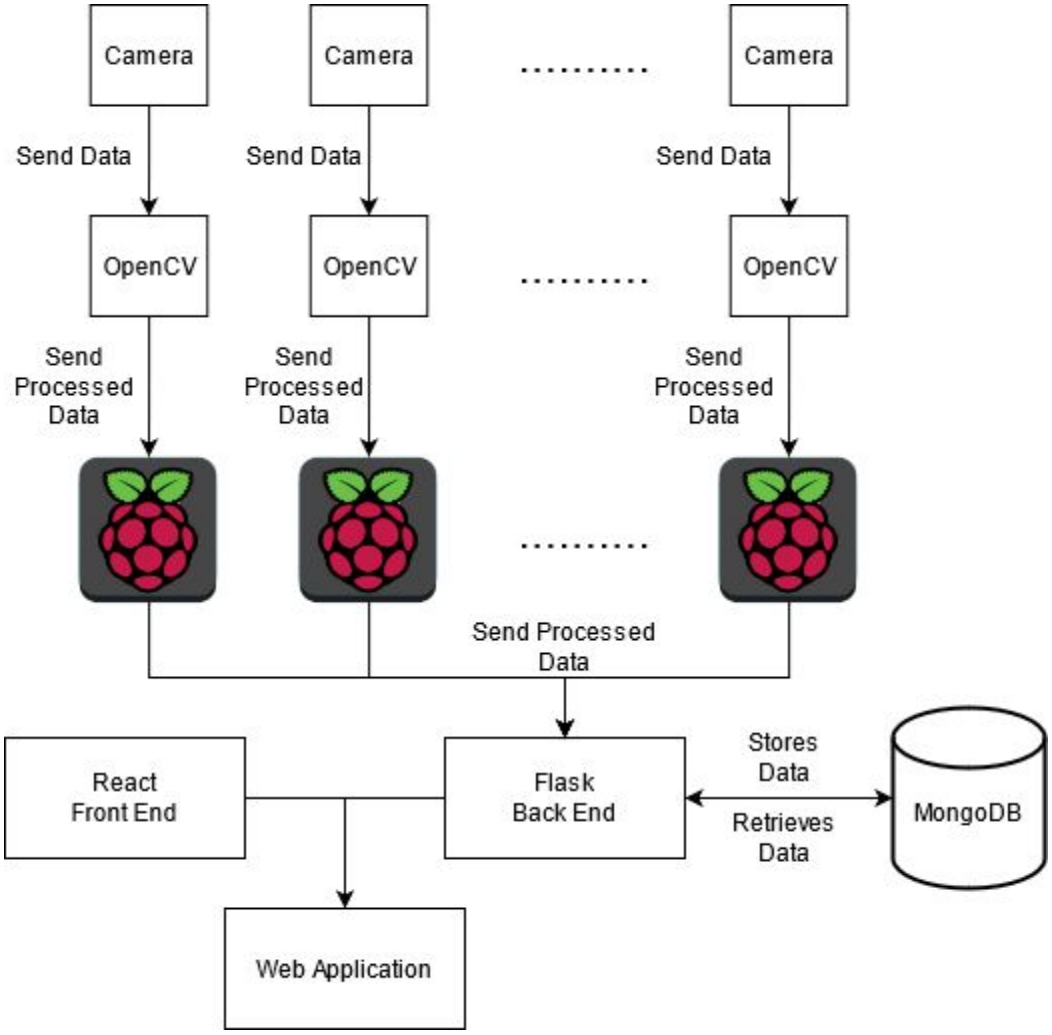

Figure 1: Architectural Overview Diagram

Our design includes eight modules. The first consists of many IoT devices connected through NAU's network. These devices do not talk to each other, however, they communicate to our backed end hosted locally at NAU. Each device will be placed in areas throughout buildings and classrooms and will have a camera mounted on them to gather data. This camera will both count people in its field of view and report that count to our backend. This counting will be done utilizing OpenCV and TensorFlow Lite. This information will be relayed to the backend with information specifically for that endpoint. This information will then be processed and delivered to our front end correctly formatted.

The front end will then display the data to the users via a timeline graph to see the recent activity in the building and/or room. This graphical representation can then be

configured by the user to determine the time interval they wish to see. This information will be stored in a database making it easily accessible to our backend to push and pull data from. The front end application will also consist of a login screen to give users different levels of access. This access will consist of admins and regular users in order for users to find what they need efficiently there will be a search feature breaking up the results into buildings and rooms where the IoT devices are currently set up and operational.

The whole structure will be discussed in depth in section 4.0 with more explanation on the design of each module.

4.0 Module and Interface Descriptions

4.1 Web Application Front End

To display the desired information to the user, we are using React to build an appealing and easy to use interface. Our frontend design consists of an object oriented programming model with the use of components and contexts. Utilizing contexts will give us the ability to quickly pull the desired data and provide an overarching connection to all of our visualization components in a non convoluted manner. Components provide the ability to use them within our chosen context fairly easily for simple data consumption and display. These components are within two main categories, data search and selection as well as data visualization.

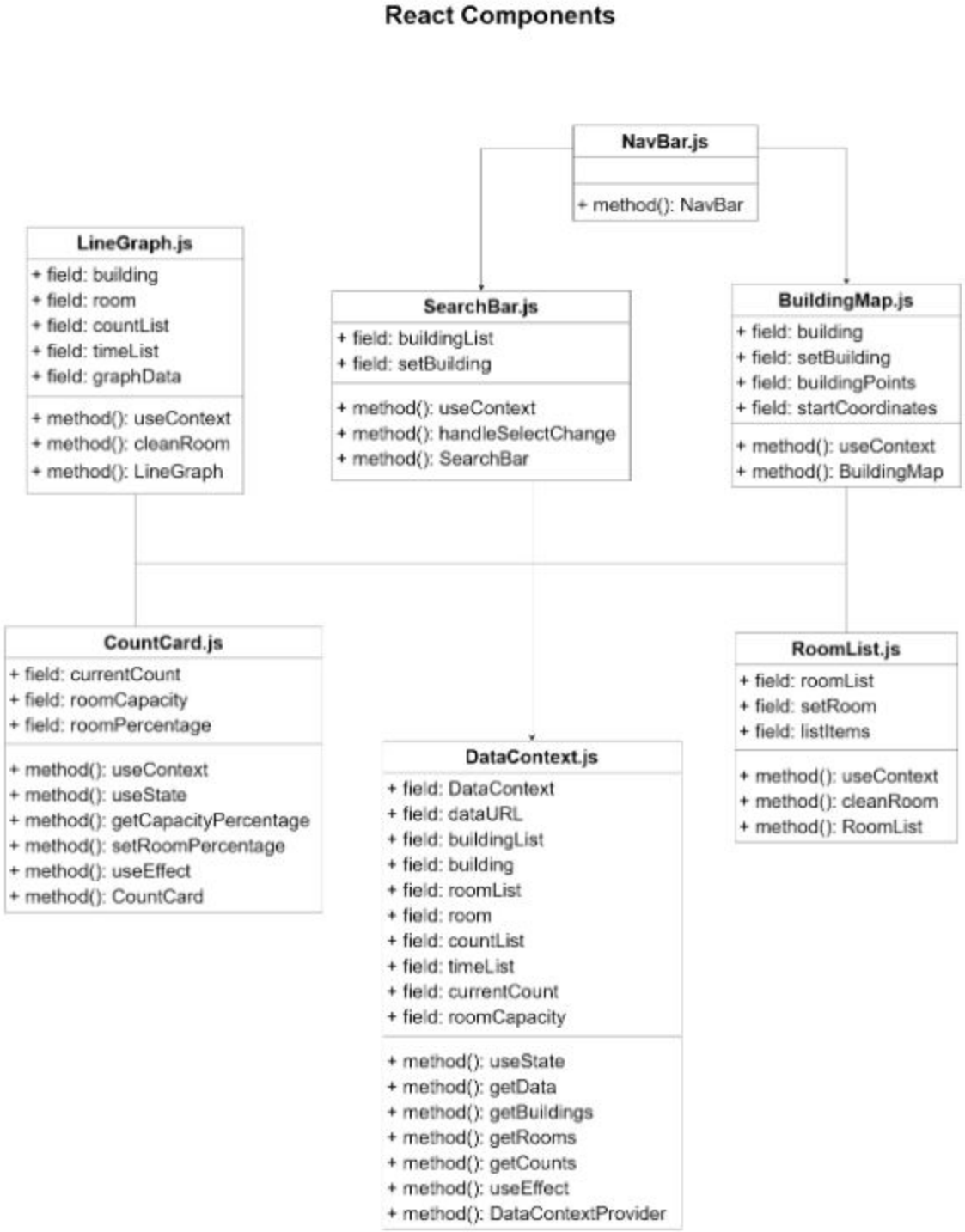

Figure 2: Front End Design Diagram

4.1.1 Data Search and Selection

Beginning with the data search and selection components. Our initial component for the user is the navigation bar, consisting of the child components, search bar and map selection. The search bar allows the user to search for and select the desired building textually. As well as the map component allowing the user to search for their building of choice visually, within a large map with zoom in and out capabilities. With a building selected in either form, a list of rooms from within the selected building will then be displayed in a scrollable format in our room list component. Each room in the list will have visual cues alongside for a quick look at room capacity and trends. The user will then have the option to select a room. On selection, the data from the chosen room will then be pulled and parsed for display within our data visualization components.

4.1.2 Data Visualization

Our data visualization components include a line graph with varying intervals for displaying room counts and trends in past or current tense. The graph will fit and change size accordingly to the user's chosen data. An accompanying infographic card component will display overall summarized data for quick statistics.

4.2 Web Application Back End

This following section describes the back end portion of our People Counting Infrastructure. The backend contains a REST API which is called People Counting Infrastructure REST API (PCI REST API). PCI REST API provides resources and endpoints to call the different services that connect the entire infrastructure together. The services that our backend provides is a Login Authentication Service (LAS) and a MongoDB Manager Service (MMS).

4.2.1 Login Authentication Service (LAS)

The Login Authentication Service (LAS) is an internal service that provides a layer of security that ensures only authorized users are able to access the web application as well as the resources/endpoints that the PCI REST API provides. This service will provide end users the ability to login or create an account to gain access to the resources that our application provides. A UML diagram of the envisioned service is shown in figure 3.

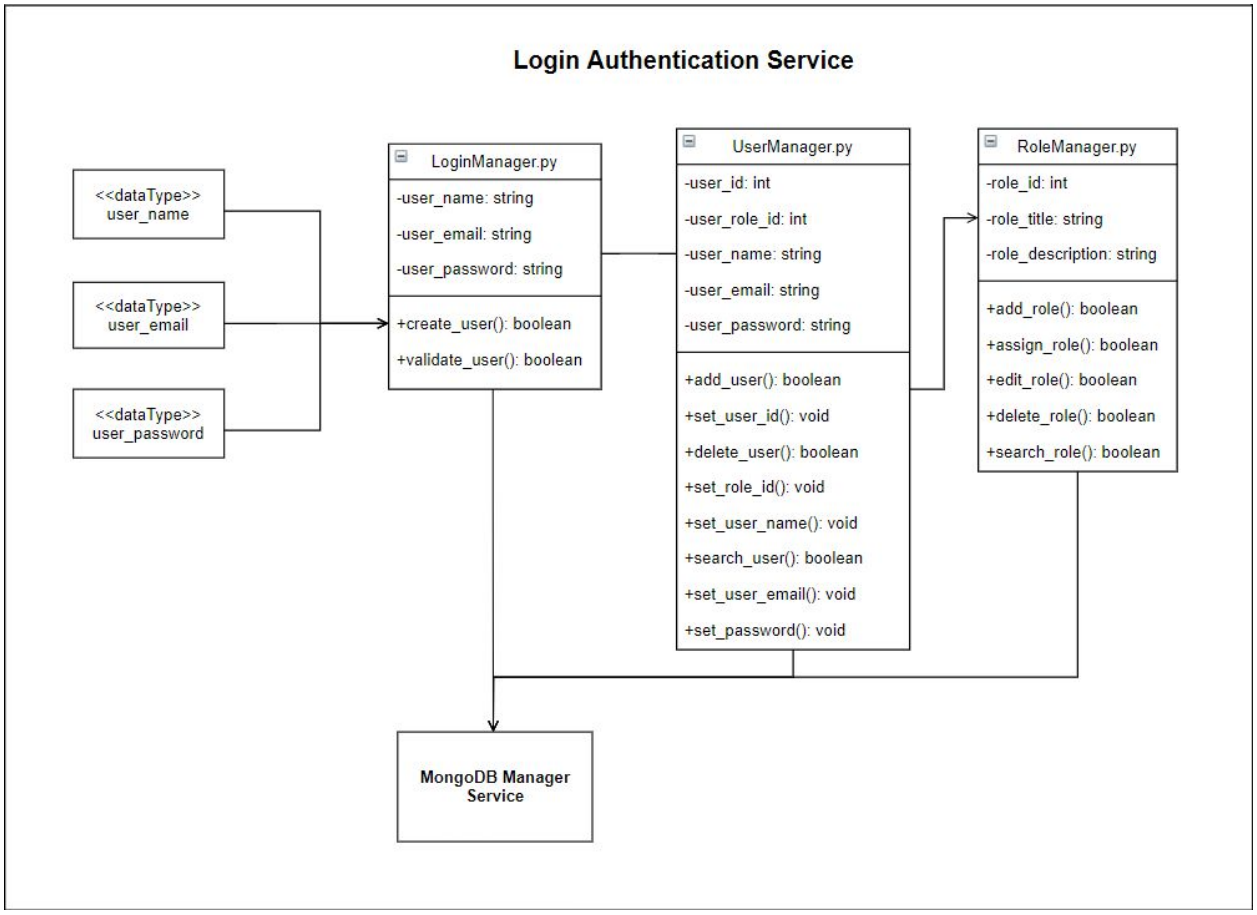

Figure 3: Login Authentication Service (LAS) Design Diagram

The front end of our web application will create a HTTP request with the following data user_name, user_email, and user_password. This data will be fed into our Login Manager, which handles creating a user and storing it into our database. If a user needs to be created, then it will interact with our User Manager to create an account based on the values that were passed into the request. The User Manager is responsible for creating a user account and storing the account information as well as the appropriate roles needed through our Role Manager in the MongoDB Manager Service (MMS). If the end user already has an account with the system, it will validate that the user exists within our application through the MMS and give the user the appropriate access to the resources needed.

4.2.2 MongoDB Manager Service

The MongoDB Manager Service (MMS) is an internal service that provides functionality for both the front end and the IoT device to gain access to our database. This service allows the IoT device to store it's entries directly into the database. This service also allows the front end to pull information from the database to create metrics based on the

count data. Another thing that this service provides is the necessary functionality needed to save user account information into the database as well as search for a user account. In figure 4 is a UML diagram of the envisioned service:

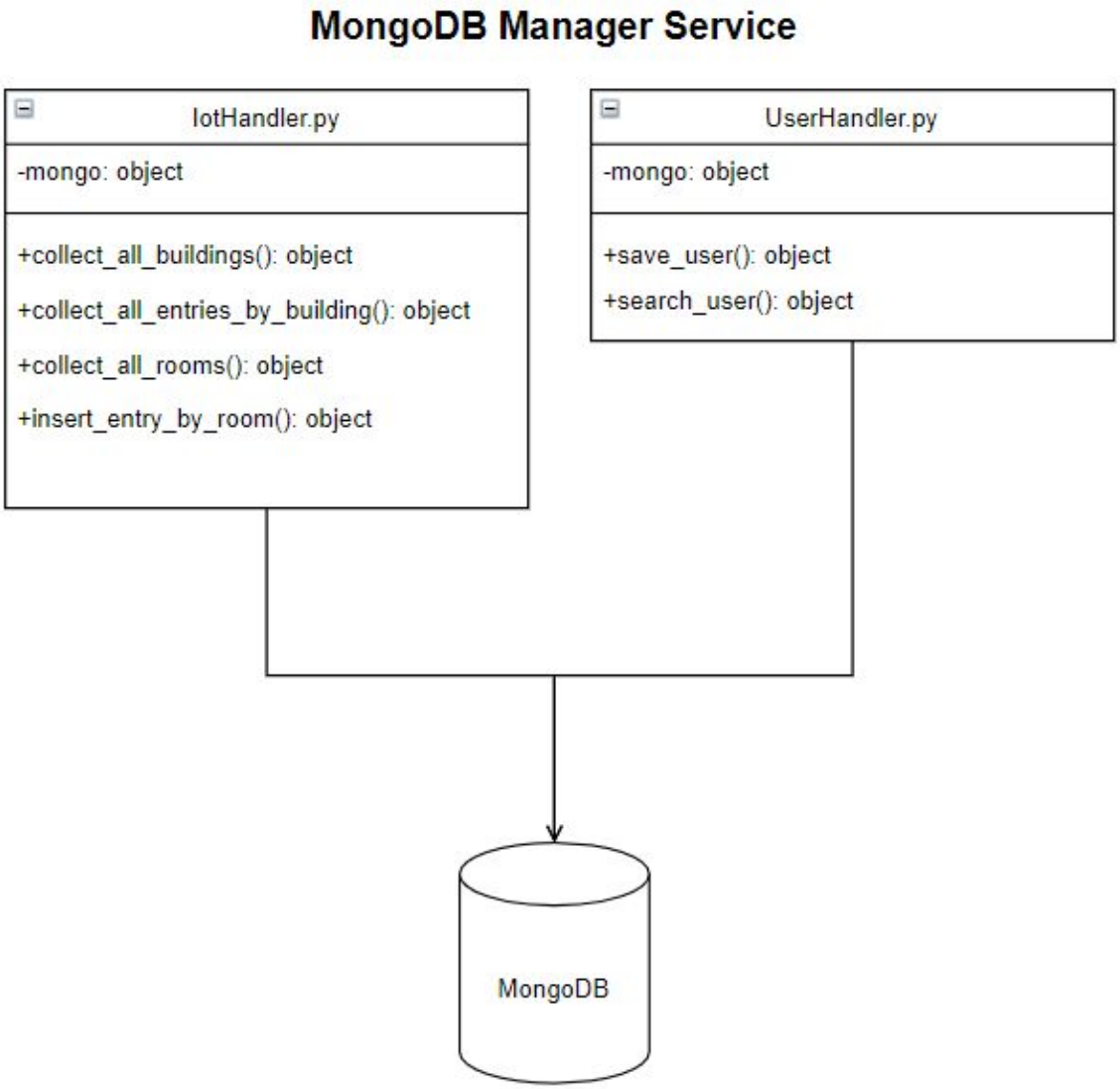

Figure 4: MongoDB Manager Service Design Diagram

As shown in the diagram above, the IotHandler gives IoT devices the ability to insert each entry by room into the database. Along with that it also provides the front end the functionality needed to collect information about the data the IoT device is gathering. The User Handler is responsible for handling the user account information that is stored

in the MongoDB database. The user handler provides the functionality for saving user account information directly into the database as well as functionality for searching if a user exists in the database.

4.3 Internet of Things Device

In figure 5 is a complete UML diagram that encompases all of the different modules within the IoT device. This section will explain how this whole model fits into the overall structure of our solution. In addition, there will be a sub section explaining each module within the diagram in more detail. Each module is split up into three sections as follows: Name, Parameters, Functions.

This model envelops the camera, OpenCV and TensorFlow Lite, and IoT Device portion of the diagram seen in Section 3.0 of this document. What the IoT device is doing is taking a video input, counting the number of people within the video via OpenCV and TensorFlow and outputting that count to our backend which in return sends it to our database. This is the first and most crucial step in getting the data to our users. Without this step we will have no data for the backend to process and our front end to show.

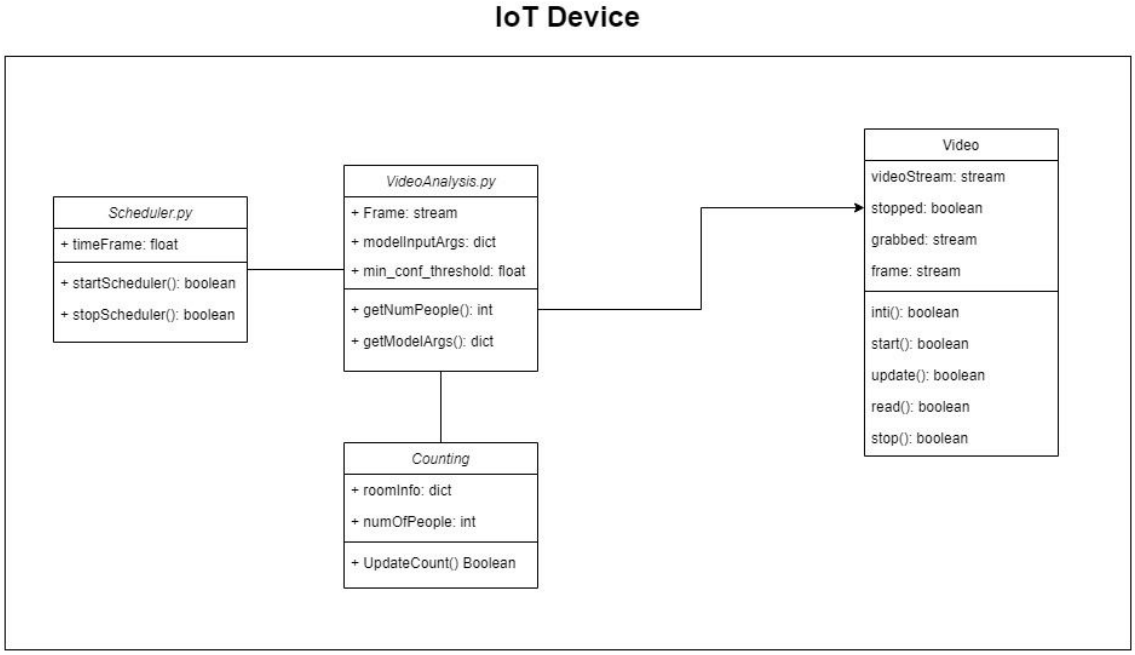

Figure 5: IoT Device Design Diagram

4.3.1 Scheduler Module

This module will be used to set a schedule on when we wanted the Video Analysis module to run. It does this by taking in a time frame and suning the Video Analysis module based on that time frame. This will be done using a cron job within python. The job will have defaulted parameters. The only thing changing will be how often it runs per day given by the number of seconds entered. There will be global constraints on the number of seconds entered to ensure the Video Analysis module has enough time to run. Once an average for that module has been determined we will make it so a time below that average cannot be entered. The stop schedule feature is simply there as a fail safe to end the cron job for whatever reason. We do not envision this to be utilized unless the admin sees fit.

4.3.2 Video Analysis Module

This is the main module within the diagram. The purpose of this module is to output a count based on the incoming video stream, threshold, and model. The threshold is used to determine the confidence in which the model given should detect a person. This is mainly for developer access so if the model is changed or improves the threshold can be adjusted as needed. The model input arguments are used to receive a pre-trained model to be utilized in the functions. This model will have to be a specific type in order to work properly with the hardware and any accelerators that we plan to use. The get number of people function will take in a Video class and model arguments to do its analysis needed to output the number of people it finds within a video frame. This function will utilize getModelArgs() in order to get the proper arguments needed from the model for python to use OpenCV properly. The model that was initially passed in will be utilized here to get the desired arguments for OpenCV and TensorFlow to use. Upon completion the counting module will be called to execute the final part of this process.

4.3.3 Video Class

This is a class that will be utilized by the Video Analysis module. This class consists of a video steam, stopped, grabbed, and frame variables. The video stream is a variable that will store the live feed of the video to be accessed in later methods. Stopped is a boolean that ensures the feed is live and has not been interrupted. Grabbed and frame are used to get a frame from the video stream. This is done utilizing a function within OpenCV. Grabbed is a boolean to tell if the frame was successfully stored. Frame is where we are storing the image to be analysed. The methods within this class are to edit, set or update said variables. Start: begins the stream, update: updates the current frame stored in the class, and read: will return said frame.

4.3.4 Counting Module

This module will execute after the Video Analysis module has returned its count of the area being read in. This module's main purpose is to send an updated count to our backend. The room info is provided as a dictionary containing strongs that will tell the function which endpoint to hit on our backend to ensure we are updating the current count as well as ensure each count is different based on what IoT device this module is being run on. The function will then return a boolean value to ensure it executed properly.

5.0 Implementation Plan

In section 4.0, our team has outlined and discussed what modules we are designing and the implementation decisions that our team has made to create the final product for our client. In this section, our team will discuss and provide the implementation timeline for our project. In figure 6, the Gantt chart displays our team's estimated timeline for the project based on the requirements and major milestones that our team needs to accomplish.

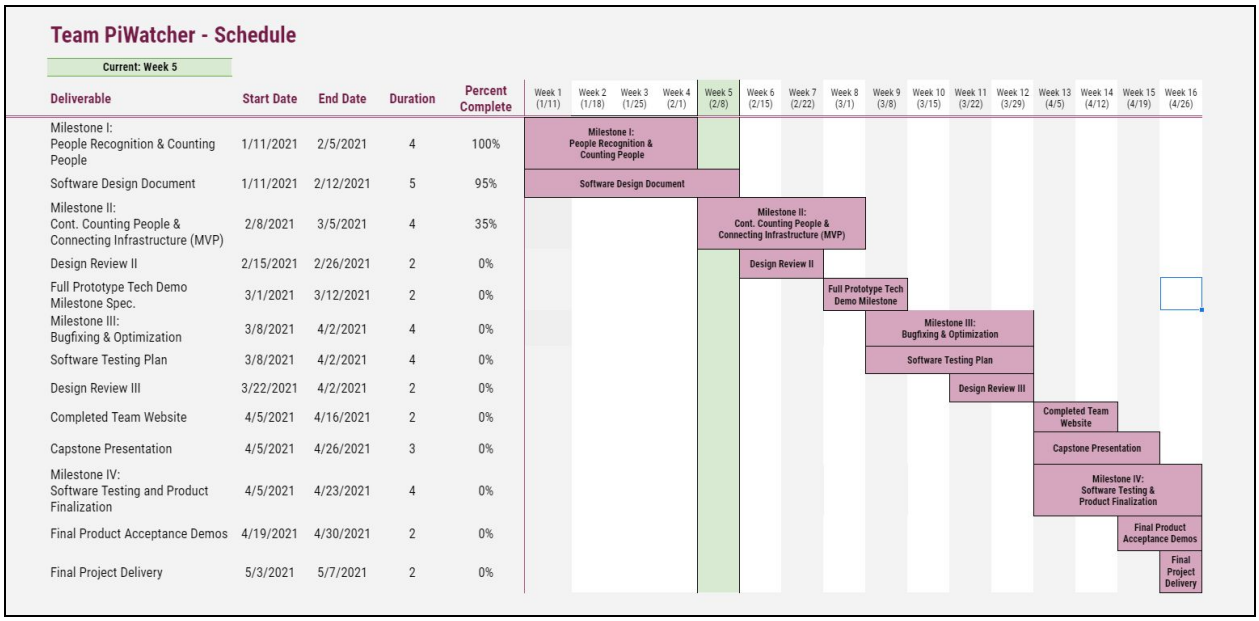

Figure 6: PiWatcher Week 5 Status Gantt Chart

During the first milestone of our project, the main focus is to get a working implementation of our people recognition module and counting module for the IoT device. In this phase, the IoT device shall be able to process feed from the camera, recognize and count how many people are present in the space that it is observing.

During the second milestone of our project, our team's main goal is to finish the development of the minimum viable product of each module of the entire infrastructure. Once the minimum viable product of each module is completed, we will start integrating the and connecting the entire system together to complete the minimum viable product of our system.

During the third milestone of our project, our team's main goal is to focus on application optimization and focus on further development. Our team will use the baseline of our minimum viable product to focus on adding additional features, fix issues and bugs, and look to optimize our people recognition and counting module to increase accuracy and maintain as well as improve performance. This phase of the project aims to meet the final requirements of our client as we move on towards software testing and product delivery.

During the fourth milestone of our project, our team's main goal is to prepare the product for product delivery. Our team will focus on implementing unit testing and integration testing, as well as needed bug fixing, final features that will be added into the final versions of our product.

After the fourth milestone, our project is done and will begin the process of formally delivering the product to our client. We will introduce the product and demonstrate how to properly use the tool as envisioned. Along with that, we will be sure to include a user manual that contains detailed information including setting up each IoT device and utilizing the metrics provided by the web application.

6.0 Conclusion

The goal of our team is to develop a people counting solution to accumulate data regarding the footfall of buildings and classrooms on NAU campus all while being cost efficient. There are many edge cases needed to be accounted for in the act of counting people directly.

During our research, our solution has come down to four issues. First, we needed to find the most capable hardware for our use case that would fit into our limited budget. We chose the Raspberry Pi, due its low cost and moderately capable processing power. Next, was finding an open source software that would work properly with our chosen hardware and provide accurate results when keeping track of people. OpenCV and Tensorflow Lite ran on the Pi and accounted for people successfully. After our first two solutions had manifested, we began to picture our accompanying web application. A modular, scalable front end framework was needed with the ability to update and

refresh in real time to display the accumulated data. We have planned our solution in React due to its high capability in our testing. In between the Pi and the application's front end is our back end, Flask. Flask handles data consumption from the Pis and is accompanied by MongoDB, our storage system. All connected, our IoT devices will count the people in view and send these counts to the back end for storage. The front end will connect to the back end and retrieve this data for viewing with authentication being handled by our own LAS system. Our solution will give an inexpensive and effective way to count people within NAU buildings and classrooms as well as provide these short and long term statistics in a business centric way.