Person Identification System

Jenna Davis, Lee Joosung, Shinya Honda, Jordan Stickley, Liping Huang

Submitted – 5/7/2018

Submitted to

Dr. Razib Iqbal

Assistant Professor of Computer Science

Missouri State University

in partial fulfillment of the

CSC450 Course Project

Spring 2018

|  |  |  |
| --- | --- | --- |
| **Person Identification System(PIS)** | | |
| Team Information | 1. Lee Joosung  2. Liping Huang  3. Shinya Honda  4. Jordan Stickley  5. Jenna Davis | |
| Document Type: Software Requirement Specification v3.0 | | *Last updated on*: 05-07-2018 |

**Document Approval**

This Software Requirements Specification has been accepted and approved by the following stakeholders:

|  |  |  |
| --- | --- | --- |
| **Printed Name** | **Title** | **Signature and Date** |
|  |  |  |
|  |  |  |
| R. IQBAL | Customer/Course Instructor |  |

Contents

[1.1 Purpose 3](#_Toc513126073)

[1.2 Scope 3](#_Toc513126074)

[1.3 Definitions, Acronyms, and Abbreviations 3](#_Toc513126075)

[1.4 References 3](#_Toc513126076)

[1.5 Overview 4](#_Toc513126077)

[2. General Description 4](#_Toc513126078)

[2.1 Product Perspective 4](#_Toc513126079)

[2.2 Product Functions 4](#_Toc513126080)

[2.3 User Characteristics 5](#_Toc513126081)

[2.4 General Constraints 5](#_Toc513126082)

[2.5 Assumptions and Dependencies 5](#_Toc513126083)

[3. Specific Requirements 6](#_Toc513126084)

[3.1 Functional Requirements 6](#_Toc513126085)

[3.2 Non-Functional Requirements 8](#_Toc513126093)

[3.3 Design Constraints 9](#_Toc513126098)

[3.4 External Interface Requirements 9](#_Toc513126099)

[3.5 Logical Database Requirements 10](#_Toc513126104)

[3.6 Other Requirements 10](#_Toc513126106)

[4. Change Management Process 10](#_Toc513126107)

[5. System Models Design 11](#_Toc513126108)

[Appendices 32](#_Toc513126109)

[Appendix I: Architectual Diagram 32](#_Toc513126110)

[Appendix II: CR Log Template 33](#_Toc513126111)

1. Introduction

This section gives a scope description and overview of everything included in this SRS document. Also, the purpose for this document is described and a list of abbreviations and definitions is provided.

## 1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the “Person Identification System” (PIS) software. It will detail all of the features to be in the final product. It will also explain hardware and software constraints, and how it would work with the different interfaces. The main purpose of this document is to be shown to the customer for approval, and then used as a blueprint for developing.

## 1.2 Scope

The Person Identification System will track unknown people from video footage by applying unique labels to tracked individuals using a deep learning neural network. It will then be able to track the person from non-overlapping video feeds to predict where they will end up.

Security personel will be able to connect to the web-based GUI from tradition desktop/laptop computers or their smart phones. The Person Identification System is designed to track persons at a local scale due to the increased criminal and terrorist activities around the world in public places such as airports, schools, and shopping centers.

Furthermore, the software needs an internet connection to display the web-based UI. All of the information is maintained on a database. The database is then read to display textual logs of information to the user on the web-based UI.

## 1.3 Definitions, Acronyms, and Abbreviations

* PIS **–** Person Identification System
* GUI – Graphical User Interface
* UI – User Interface
* API – Application programming interface
* NN – Neural Network
* IP Address – Internet Protocol Address
* DB – Data Base
* VideoController – the subcomponent of this applciation that interfaces with the usb camera provides video streaming back to the WebView
* WebView – The main browser based UI for this application.
* CR – Change Request

## 1.4 References

1. ieeexplore.ieee.org/iel7/8234942/8237262/08237573.pdf
2. ieeexplore.ieee.org/document/8253595/
3. ieeexplore.ieee.org/iel7/7963867/7973339/07973347.pdf
4. <https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/>
5. <http://www.chioka.in/python-live-video-streaming-example/>
6. <http://flask.pocoo.org/>
7. <http://flask-mysqldb.readthedocs.io/en/latest/>
8. <https://dev.mysql.com/doc/mysql-installation-excerpt/5.7/en/>
9. <https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html>
10. http://shervinemami.info/shirtDetection.html

## 1.5 Overview

The remainder of this document includes three chapters and one appendix that detail the requirements and constraints surrounding the PIS. Section two provides an overview of the general system functionalities. Section three provides the requirement specifications in detailed terms and a description of the different system interfaces. Section four indentifies and describes the process that change managerment of the project, and when the project scope or requirments change. It also outlines who can submit changes and by what means. The appendix contains diagrams that can assist with visualization and understanding of the project.

# 2. General Description

This section will give an overview of the system. The general workings of the system and basic functionality will be introduced. The potenial users of the system will also be descibed in this section. The constraints and assumptions for the system are introduced here as well.

## 2.1 Product Perspective

The application is designed to label a person and track them in a public environment. A camera tracks an unknown person and analyzes the unknown person. By means of a NN, the camera frames will identify the unknown persons. This data will be strored in a database and the application will analyze and remember the person by assigning them a unique identifier. If another camera in the system recognizes the same person, the application will identify that person as the person from the database. If a new person enters the view of the cameras, they will not be recognized as a person that is already in the database and will be added as a new database entry, see appendix I for architectua diagram of the system.

## 2.2 Product Functions

The system will consist of a series of cameras connected to computers. There will also be a main computer to host the web-based UI and the database. The computers that run the video algorithms will then connect to the main computers database by means of IP address. The database will be used to store persons identified by the system and tracking information. See Figure 1, Appendix I for a diagram of the system’s layout.

The system will automatically identify a person as human and perform a series of operations on the camera image to retrieve identifying values from that person. These identifying values will be stored in a database and used to uniquely identify and create a label for that person.

The system will be capable of recognizing that a person whom was previously identified is the same person between non-overlapping video feeds. It will also be able to recognize a ‘new’ person and will add them to the database instead of considering them as a previously labeled person.

The system will also work to predict where a person is going and what area they will appear in next. Therefore, the software will work to identify unique persons, track those people between non-overlapping video feeds, and predict their destination within the system.

## 2.3 User Characteristics

The primary users of this software are intelligence agencies and security organizations such as those seen at airports and casinos.

Other responsibilities will be recognizing the systems movement detection notifications and responding appropriately; such as clicking the camera links to view the video feed. The individuals being tracked will likely all be unknown to the users of the software, thus all identifying features must be gathered by the system. A mechanism for identifying and tracking users of the software is also an important feature as the software will contain surveillance data of public spaces and a record of the people whom have access to that data should be recorded. Since one of the responsibilites of the users is to call for back up in case of an emergency, they will need a way to predict where the subject might be heading so as to more efficiently dispatch assistance.

At this point in the surveillance it is more important to recognize and label each unique person for tracking purposes than to recognize them as a specific person thus the subjects will not be identified but instead classified by identifiers that the system will pull from their image. For example, the system will recognize that a person is ‘Person A123’ between multiple cameras instead of recognizing that the person is ‘John’.

## 2.4 General Constraints

The system will use a network of ELP Mini USB Cameras for the purpose of viewing video feeds and applying the tracking algorithms to image frames.

The computer being used is another constraint of the system. The hardware that executes the image processing algorithms must have at least 2.4 GHz Intel core i3 or higher; the algorithm may not run at optimum speeds which could slow down the overall calculations. This could lead to low system performance or algorithm failure.

A strong Internet conenction is also a constraint for the application. Because the program retrieves data and video streams over the internet, it’s imperative that there be a strong connection so as to have a clear view of the feed and minimal delays.

2.4.1

## 2.5 Assumptions and Dependencies

One assumption of the system is that the computer that is executing the algorithm has at least a 2.4 GHz Intel core i3 or higher to properly run the algorithm. If the system is being ran on less than this, the performance will be decreased as it will not have enough processing power to handle the video feeds and run the software.

Another assumption is that the camera placement of the system will be appropriate to provide proper information. If there are too few cameras it will be difficult to get a good view of the location the system working in, and the subject being identified may not walk into the view of more than one camera. This would also make tracking the subject more difficult for the system as patterns would be more difficult to recognize.

# 3. Specific Requirements

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

## 3.1 Functional Requirements

### 3.1.1 A user shall be able to see how many cameras are configured from the main view.

**Source:** Team Discussion.

**Priority:** 1

**Introduction:** All known cameras are displayed.

**Inputs:** Camera data read from the database.

**Processing:** Loading camera information from the database and rendering as a list.

**Error Handling:** Display message when couldn’t communicate with database.

### 3.1.2 A user shall be able to click links to see each camera’s feed.

**Source:** Team Discussion.

**Priority:** 2

**Introduction:** The per-camera view can be seen.

**Inputs:** A click on the link.

**Processing:** React to the click event and render the video stream from the selected camera.

**Outputs:** Display the video feed.

**Error Handling:** Camera unavailable displayed.

### 3.1.3 The Person Identification System shall display a label and a tracking rectangle on moving people.

**Source:** Customer meeting.

**Priority:** 3

**Introduction:** 1. A box is drawn around moving people, and is labeled appropriately.

**Inputs:** Video.

**Processing:** Person detection algorithm based on neural net identifies and returns coordinates for drawing the bounding box.

**Outputs:** Display the lable and the rectangle enclosing moving people.

**Error Handling:** None.

### 3.1.4 The user shall be able to see the camera label and neighboring cameras from the main camera view screen.

**Source:** Team Discussion.

**Priority:** 4

**Introduction:** Neighboring camera links are available to click and the specific camera label is in view.

**Input:**Camera identification information that is read from the database.

**Processing:** None.

**Outputs:** The active camera label and hyperlinks for neighboring cameras.

**Error Handling:** None.

### 3.1.5 The movement tracking system shall take control of the camera and will capture person movement information into a database.

Source: Team Discussion.

**Priority:**5

**Introduction:** Camera can be activated, movement data can be detected, and tracking information can be stored in a database.

**Input:**Video Data]

**Processing:**

**Output:** None.

**Error Handling:** A message is displayed showing that Camera data was unable to be communicated with the database.

### 3.1.6 The movement tracking system shall label a tracked person that’s moving into a later camera with their original label.

**Source:** Team Discussion.

**Priority:** 6

**Introduction:** Original label is rendered in new camera.

**Input:** Tracking information from database.

**Processing:** Seach information of the person in the database, if found, label them with that information.

**Output:** Original label on tracked person who has moved into a new camera.

**Error Handling:** None.

### 3.1.7 The movement tracking system shall operate autonomously on all camreas, storing their data in a shared database.

**Source:** Team Discussion.

**Priority:** 7

**Introduction:** System accommodates new cameras as they come on line without up front pre-configuration.

**Input:** None.

**Processing:** A database and the main web view are able to react to cameras as they are brought online or taken offline.

**Output:** Previously offline cameras are now online and previously non-existent cameras are now online

**Error Handling:** None.

## 3.2 Non-Functional Requirements

### 3.2.1 A user shall be able to see movement indicators on the main screen.

**Source:** Team Discussion.

**Priority:**1

**Introduction:** Idle cameras change to indicate movement and back to idle again as per activity from cameras.

**Input:**Video Data

**Processing:** Camera list view is updated periodically to reflect indicators and requires reading this information from the database.

**Output:** Indicator that movement has occurred on a given camera.

**Error Handling:** None

### 3.2.2 The user shall be able to see a textual log of movement activity.

**Source:** Team Discussion.

**Priority:** 2

**Introduction:** Movement activity can be seen in real time and the following information can be seen: id, begin time stamp, end time stamp, origin camera id, predicted destination camera id

**Input:**Video Data

**Processing:** The textual log entries are periodically loaded from the database.

**Output:** Log entries are displayed on the webpage.

**Error Handling:** A message is displayed showing that Camera data was unable to be communicated with the database.

### 3.2.3 The movement tracking system shall track multiple people at the same time.

**Source:** Customer meeting.

**Priority:** 3

**Introduction:** Each tracked person has a unique label and they are successfully tracked as per any other movement.

**Input:** Tracking information from database.

**Processing:** Database analysis of the tracking data.

**Output:** Different database entries for each unique person being tracked as well as on-screen labeling.

**Error Handling:** None.

### 3.2.4 The Person Identification System shall be able to work with a variable amount of cameras.

**Source:** Customer meeting.

**Priority:** 4

**Introduction:** The PIS will be able work with a variable amount of cameras.

**Input:** Remote camera system and correspoding video controller software.

**Processing:**Detect when new cameras come online.

**Output:** Updated camera list in main web view.

**Error Handling:** Message is displayed showing that a camera was unable to connect properly.

## 3.3 Design Constraints

**Programming Language:**

Python and its public libraries are used for the implementation of The Person Identification System.

**Operating System:**

The Person Identification System is designed to be developed and worked on with Microsoft Windows 10 (32bit/64bit), Linux, and OS X operating systems. Working on older operation systems and any other operating systems is not a concern of this project.

**Platform:**

The Person Identification System is designed to work on any Personal Computer or Smart Phone Device. Working on any other devices is not in concern of this project.

**Hardware limitation:**

Cameras and computers will be provided by team members. The Person Identification System can take any playable recoded video or streaming video as input data. However, person identification is totally reliant on quality of video data such as resolution, color, light etc. The number of people and the number of video/cameras that The Person Identification System can handle at the same time will rely on performance of hardware it runs on – lower performing hardware may reduce the systems performace and could cause the tracking function failure.

## 3.4 External Interface Requirements

### 3.4.1 User Interfaces

3.4.1.1 The Person Identification System(PIS) shall provide a web-based UI for user to interact with the software.

**Source:** Team Discussion.

**Priority:** 1

**Introduction:** Upon accessing the application with their browser, the user can see the main view.

**Inputs:** None.

**Processing:** None.

**Output:** Web page.

**Error Handling:** Display relevant message when no data exists.

### 3.4.2 Hardware Interfaces

3.1.2.1 The Person Identification System(PIS) shall access cameras through usb port.

**Source:** Team Discussion.

**Priority:**1

**Introduction:** The VideoController component of this application interfaces with a usb camera using the cv2 python library.

**Inputs:** None.

**Processing:** None.

**Output:** Not Applicable.

**Error Handling:** Display relevant message when no data exists.

### 3.4.3 Software Interfaces

3.4.3.1 The Person Identification System(PIS) shall provide a Flask web application framework UI.

**Source:** Team Discussion.

**Priority:**1

**Introduction:** The Flask web application framework is used as a foundation for the UI and the video streaming component of the VideoController

**Inputs:** None

**Processing:** None.

**Output:** Web page.

**Error Handling:** Display relevant message when no data exists.

### 3.4.4 Communications Interfaces

3.4.3.1 The Person Identification System(PIS) shall connect to Mysql database through standard Mysql socket protocols.

**Source:** Team Discussion.

**Priority:**1

**Introduction:** Mysql database is communicated with from the WebView and VideoController components using standard mysql socket protocols

**Inputs:** None.

**Processing:** None.

**Output:** None.

**Error Handling:** Display relevant message when no data exists.

## 3.5 Logical Database Requirements

### 3.5.1 The Person Identification System shall use a database for its storing attributes of people.

**Source:** Team Discussion.

**Priority:**1

**Introduction**:

**Inputs**:Attributes

**Processing**: Not Applicable

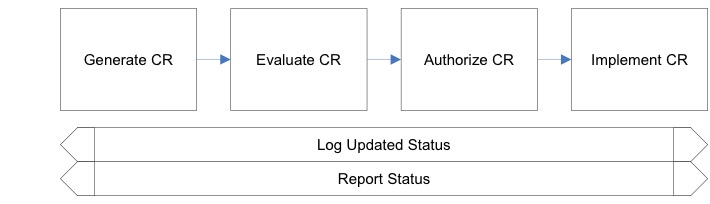
**Outputs**: Not Applicable

**Error Handling**: It shall handle missing attributes by using null value.

## 3.6 Other Requirements

Not applicable

# 4. Change Management Process

This chapter presents our teams practices in change management. It is intended to be a step-by-step process for changing various aspects of our project’s initial design. This is so that all parties are satisfied with the proposed change and can be recorded appropriately. The follow figure shows what change management process looks like:

If a CR needs to be generated, the first step is to fill out a short description of what will be changed and it will also include expected start and end dates for finishing the change, what type of change it is (software, hardware, other), and how difficult and how important the change is.

The second step is to have our current scrum master record the CR on Trello. The CR will then be available for updating throughout the CR process. In Appendix II, a template for the log can be seen.

In step three, we will evaluate our CR in a team meeting with our customer and stakeholders. During this step, we will review the CR and provide an estimated level of effort to develop a proposed solution. We will use a important vs difficulty comparison to help decide whether or not the time investment is worthwhile. For any CR with importance greater than difficulty should be approved. The importance and difficulty will be calculated as a vote during team meetings with stakeholders.

Authorization of the CR is the fourth step in the process. In this stage we will either approve or reject the CR based on our evaluation from step 3.

The fifth and final step is to implement the CR. If approved, we will make any necessary adjusments to carry out the request change and communicate the CR status to the scrum master and other stakeholders.

# 5. System Design

5.1.0 Introduction

The purpose of this software design section is to give a low-level description of the Person Identification System and to outline the design of the project. The following topics will be covered:

* Data design.
* Architectural and component-level design.
* User interface design.
* Restrictions, limitations, and constraints.
* Testing Issues.

After reading this document, you should have a good understanding of how the Person Identification system functions.

5.1.1 Goals and objectives

The purpose of the Person Identification system is to be able to label unknown people with a unique “code”, and then be able to re-identify them from a non-overlapping camera that’s connected through a shared database. We also aim to provide destination prediction system that will function with multiple tracked people being tracked by the system at the same time. So, the primary objective is to create a functioning tracking and identification system.

This product must be able to accurately label and track unknown individuals quickly and efficiently. The system must be easy to setup. The user interface must be easy to understand and provide efficient navigation to each camera view. Other than these general design requirements, the application must also do the following functionalities:

* Web portal accessible through LAN.
* Work with up to 5 cameras set up.
* Communication through database amount camera side program.

5.1.2 Statement of scope

The Person Identification System runs using two main systems: the user interface that both receives user input and outputs the camera feeds, and a server-side database that communicates information throughout the camera network and updates the textual log. We will then break our features into three main groups: essential, which are mandatory to the basic functions of the system, desirable, which are additional features which we hope to complete, but are not sure about, and future requirements, which we have strong doubts about.

*Essential Features*

1. Access Main UI

**Input**: URL to access hub.

**Processing**: Loading camera information from the database and rendering as a list.

**Output**: Video hub is successfully displayed.

1. Video Select

**Input**: Click camera link.

**Processing**: React to the click event and render the video stream from the selected camera.

**Output**: Selected camera’s video feed is shown.

1. Motion Indicator

**Input**: Person walks into view of active camera.

**Processing**: Camera list view is updated periodically to reflect indicators and requires reading this information from the database.

**Output**: Motion indicator lights up.

1. Person Detection

**Input**: Person moves in into camera view

**Processing**: Person detection algorithm based on neural net identifies and returns coordinates for drawing the bounding box.

**Output**: Person is surrounded by labeled box

1. Multiple Tracking

**Input**: Person 1 and Person 2 each enter camera views, Person 1 and Person 2 then each move to a different camera.

**Processing**:

**Output**: Person 1 and Person 2 both have correct motion and prediction indicators display on their respective cameras, also they are both correctly identified and labeled.

1. Textual Log View

**Input**: Activity occurs on camera view.

**Processing**: The textual log entries are periodically loaded from the database.

**Output**: Log is shown in Activity section of UI.

1. Database connection

**Input**: Start the server application and cameras.

**Processing**: A database and the main web view are able to react to cameras as they are brought online or taken offline.

**Output**: The server application and cameras can communicate with the database.

1. On/Off Button Functionality

**Input**: On a connected camera, click the on/off button.

**Processing**: Instruction to turn on/off a connect camera.

**Output**: The camera feed has stopped/resumed in accordance with the on/off click.

*Desirable Features*

1. Destination Prediction

**Input**: Tracked person leaves one camera and enters a different camera.

**Processing**: Prediction base on activity log of cracking person.

**Output**: Tracked person is identified as the same person and is reassigned the label they were given from the first camera.

1. Re-Identification

**Input**: Tracked person leaves one camera and enters a different camera.

**Processing**: Search information of the person in the database, if found, label them with that information.

**Output**: Tracked person is identified as the same person and is reassigned the label they were given from the first camera.

*Future Requirements*

1.      Facial identification

**Input**: A person’s face is seen on one of the cameras.

**Processing**: A facial recognition algorithm is running on the face.

**Output**: The person is identified by their facial features, or they are stored in the database if their face is not found.

2.      Cameras and web server connection with cloud database.

**Input**:  Camera video feeds.

**Output**: information is accessible to users of the cloud.

3.      Internet accessibility   
**Input**: Cameras stream video to a secure website.

**Output**: Information is available to users via a secure website from any location.

5.1.3 Software context

The Person Identification System will be a software package that the customer can purchase through major retail sources. Due to our high-skill based team, ample funding will be necessary to the design process of this advanced system. We are, however, prepared for the case where our funds are insufficient and if such a case were to arise, we would utilize an additional monthly membership fee on top of the software’s initial purchase price. We have also envisioned a micro-transaction system where the customer can pay for additional features such as increased camera count or larger textual log display.

5.1.4 Major constraints

The primary constraint for the Person Identification System is the allotted time for completion. There is a total of about three months from start to finish in which this time is devoted to development, testing, and documentation of this project. Also, the entire team has almost no experience with working on such a project, so a lot of time will be spent learning how to appropriately work together and allocate our resources effectively. Finally, every team member has a significant work load alongside this project, so the combination of little experience and a tight work schedule could result in less features on the initial release. This should have no impact on our core design features.

5.2.0 Data design

This section describes various data structures important to the Person Identification system.

5.2.1 Internal software data structure

The person Identification System’s internal structure is divided into three parts: shared, video controller, and a web view.

*Shared*

* Class ActivityDbRow
  + Models the activity of one row of the tracking
    - id
      * unique id for each activity record written to the database.
    - Label
      * Attached to identify people.
    - Start\_time
      * When they were first identified in a given camera.
    - End\_time
      * When they went out of view of that camera.
    - Camera\_id
      * Which camera they were in when this record was created.
    - Next\_camera\_id
      * The predicted camera that they were headed to.
    - Rect\_start
      * The initial rectangle position when the record was first created.
    - Rect\_end
      * The final rectangle position when they left the frame.
* Class CameraDbRow
  + Models the activity of one row of the camera table
    - id
      * unique database id for each camera.
    - ip
      * the ip address of each camera on the network.
    - Left\_camera\_id
      * The id of the camera to the left.
    - Right\_camera\_id
      * The id of the camera to the right.
    - Is\_online
      * True or false if designating of the camera is online.
    - Has\_motion
      * True or false if the camera is currently detecting motion.

*Video Controller*

* Class VideoCamera
  + Encapsulates the communication with the physical camera.
    - cameraDetails
      * Contains a reference to the camera database row.
    - Mysql
      * Reference to database access class.
    - shutItDown
      * A Boolean value of true if camera is shut down.
    - Camera
      * A reference to a CV2 video capture reference.
    - Net
      * A reference to the neural net.
    - Jpeg
      * A reference to one from of video.
    - Capturing
      * Set to true if the camera is currently capturing frames.
    - Lock
      * A thread-lock for controlling access to jpeg reference described above.
    - tracked\_list
      * A list of the tracked people.
    - insertActivity
      * A method for inserting detected activity in the database.
* Main flask application
  + Entry point for the flask web user interface for the Video Controller
    - updateDetailsInDb
      * Records camera details upon startup of the video controller.
    - shutdownServer
      * A method for shutting down the video controller.
    - shutdownCamera
      * Shuts down the physical camera. (can be restarted)
    - checkCamera
      * Checks the connectivity to the camera and starts recording.
    - Shutdown
      * The method connected to flask that allows an HTTP request to initiate the shutdown of the server.
    - Index
      * Method connected to flask for serving up a test view of the camera.
    - Gen
      * A private method for generating one frame as a JPEG
    - video\_feed
      * The method connected to flask that allows for an HTTP request for one frame of video.

*Web View*

* Main flask application
  + Entry point for the flask user interface for the main web view
    - getCameraList
      * Method for retrieving the list of cameras.
    - getCameraListWithMotion
      * Gets a list of the cameras that actually have motion.
    - Index
      * The method connected to flask for serving up the main web view.
    - View\_camera
      * The method connected to flask for serving up the view of a single camera.
    - Home
      * The method connected to flask for serving up the home link.
    - Poll\_for\_status
      * Contains logic to return the activity logs and update the camera view.

2.2 Global data structure

* mysql
  + Reference to the database accessor for talking to Mysql.
* Camera
  + Reference to the videoCamera described above.
* cameraDetails of the videoController
  + Reference to the cameraDetails datastructure.

2.3 Temporary data structure

The Person Identification System contains temporary data structures in the form of data objects that exist in the VideoController and WebView applications. The data objects on these applications exist while the system is running and are deleted when the system is shut down.

5.2.4 Database description

The name of the database is securdb and contains two tables.

*Tables*

camera (

id int primary key,

camera\_IP int,

left\_cam\_id int,

right\_cam\_id int,

is\_online char(1)

);

tracking (

id int primary key,

label varchar(50),

start\_time timestamp default current\_timestamp,

end\_time timestamp,

camera\_id int,

next\_camera\_id int,

foreign key (camera\_id) references camera(id),

foreign key (next\_camera\_id) references camera(id)

);

5.3.0 Architectural and component-level design

For the Personal Identification System, we have 5 cameras and each camera will have its own computer to run its Video controller program. An image processing (neural network) also takes place in the camera control program. All five computers with connected cameras communicate through a database which runs on a sixth computer as the Database server. The web server is also run on this computer, which generates the web view, camera view and activity log that will be shown on the user side of the web browser.

5.3.1 System Structure

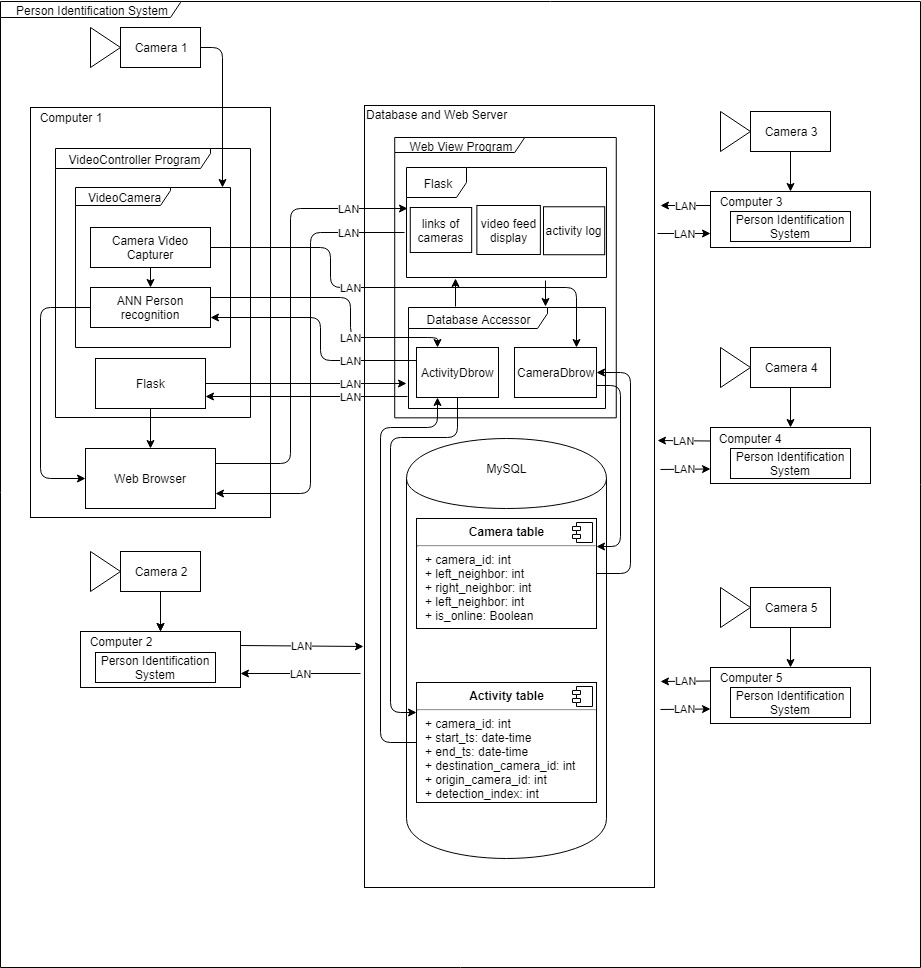
**RW Pattern**

The RW pattern is chosen to be the system’s design pattern for the Personal Identification System. We will have five different Video Controller Programs which run on their own computers. Web access by users will trigger, read and write to the same database, therefore a read and write lock is required for any read and write activity. When CameraDbrow and ActivityDbrow of the web server program is reading from the database, a write lock will prevent any other database write requests from writing to the database and the write lock is released when reading is finished. When CameraDbrow and ActivityDbrow are writing to the database, a read lock will prevent any other database read requests from reading the database and the read lock is released when writing is finished.

**Client-Server Pattern**

The Personal Identification System is designed and built on a client-server pattern. Any computational device with a connected camera, which runs the Video Controller Program can be a client. New clients can be added to the system as needed (5 are used in this project). A computer or server runs MySQL database and the Web View Program is also needed. All communication are sent through the same local area network (LAN).

5.3.1.1 Architecture diagram



5.3.2 Description for Component n

In the Person Identification System, there three major components.

1. Video Controller Program

This is the main program that runs on client side of the system, a computer with its camera.

1. Database

A MySQL database runs on the server side of the system.

1. Web View Program

A web server runs on the server side of the system.

* + 1. Processing narrative for component n
       1. Camera Controller Program

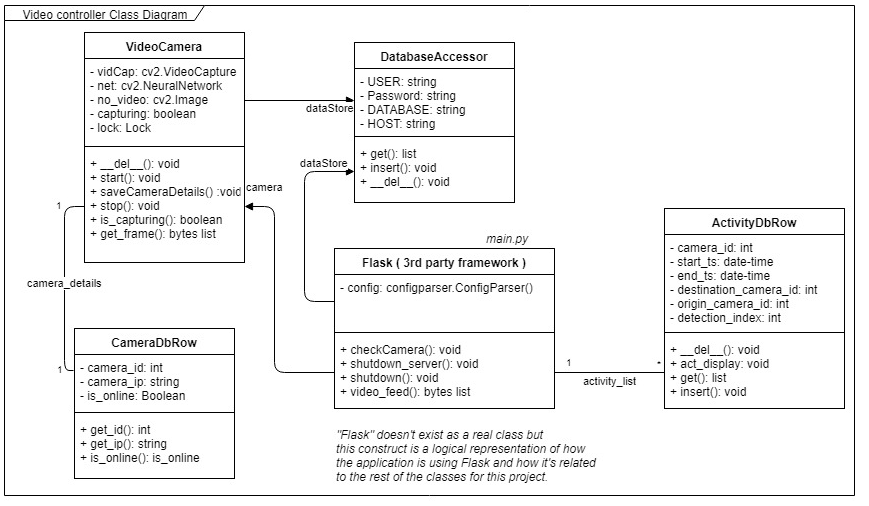
Takes camera video as input and outputs information to store into the database.

* VideoCamera
  + Camera capturer
  + NN model
  + CV2 Face recognition
  + Human body feature identify algrithm
* Flask Application
  + Video feed
  + Activity indicator
    - 1. Database

A MySQL database with two tables storing all the information for the system.

* Camera table
* Activity table
  + - 1. Web View Program

It takes information from the database and generates a web view for the user side of the web browser.

* DatabaseAccessor
  + CameraDbrow
  + ActivityDbRow
* Flask Application
  + Link of cameras
  + Video feed
  + Activity log
  + Activity indicator
    - 1. Class Diagram
    1. Component n interface description.
       1. Camera Controller Program

Takes camera video as input and outputs information to store in to the database.

* VideoCamera
  + Camera capturer

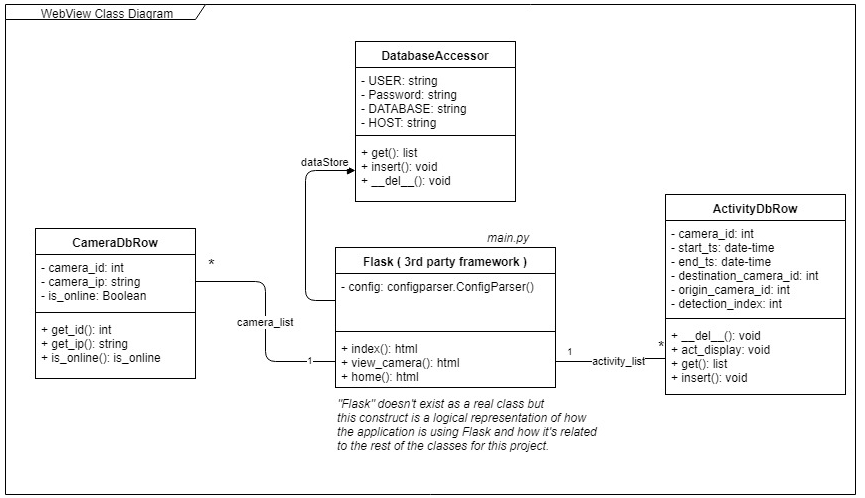
Convert video to image frame.

* + NN model

A person recognition NN algorithm take image as Input and output list of confidence and object index of an object list.

* Flask Application

A user interface development system that is used to develop the web base user interface.



* + - 1. Database

A MySQL database with two tables storing all the information for the system.

* Camera table

Storing information to keep track of all of the cameras connected to the system.

* Activity table

Storing information to keep track with all activity of people in view cameras view.

* + - 1. Web View Program

Takes information from the database and generates a web view for user side of web browser.

* DatabaseAccessor
  + CameraDbrow

Reader and writer row by row to the Camera table of the database.

* + ActivityDbRow

Reader and writer row by row to the Activity table of the database.

* Flask Application
  + Link of cameras

User interface that shows a link of all cameras connected to the system with their status icons.

* + Video feed

User interface display that shows a video feed from a camera and links to the next (right) camera/previous (left) camera.

* + Activity log

User interface shows the ten most recent activities of all cameras in the system.

5.3.2.3 Component n processing detail

5.3.2.3.1 Design Class hierarchy for component n

* Video controller Program
  + VideoCamera
  + Flask Application
* Database
* Camera table
* Activity table
* Web View Program
  + Link of cameras
  + Video feed
  + Activity log

5.3.2.3.2 Restrictions/limitations for component n

* Video controller Program

A public source NN is used in this program, but we have no access to modify the NN.

* Database

None

* Web View Program

None

5.3.2.3.3 Performance issues for component n

* Video controller Program

A public source NN is used in this program, performance may become an issue depending on the hardware.

* Database

None

* Web View Program

None

5.3.2.3.4 Design constraints for component n

* Video controller Program
* Python
* Open CV
* Database
* MySQL
* Web View Program
* Python
* Flask

5.3.2.3.5 Processing detail for each operation of component n

5.3.2.3.5.1 Processing narrative for each operation

* VideoCamera
  + Encapsulates the communication with the physical camera.
    - Del()
      * Release a camera.
    - getNextDBId()
      * Get ID of next (right) camera.
    - insertActivity()
      * Insert an activity to activity table in database.
    - saveActivity()
      * Save current camera activity.
    - getPredictedCameraId()
      * Get the ID of a camera that a tracked person might show up on next.
    - Start()
      * Camera video capture and runs the NN.
    - find\_closest\_tracked\_activity()
      * Find the closest tracked activity in activity table.
    - went\_left()
      * Current tracked person went to left camera from current camera.
    - went\_right()
      * Current tracked person went to right camera from current camera.
    - Distance()
      * Calculate the distance between two points in an image frame.
    - Stop()
      * Assign True to shutItDown
    - is\_capturing()
      * Return true if camera is capturing video.
    - get\_frame()
      * Get frames of the captured video.
* Main flask application
  + Entry point for the flask web user interface for the Video Controller
    - updateDetailsInDb()
      * Records camera details upon startup of the video controller.
    - shutdownServer()
      * A method for shutting down the video controller.
    - shutdownCamera()
      * Shuts down the physical camera. (can be restarted)
    - checkCamera()
      * Checks the connectivity to the camera and starts recording.
    - Shutdown()
      * The method connected to flask that allows an HTTP request to initiate the shutdown of the server.
    - Index()
      * Method connected to flask for serving up a test view of the camera.
    - Gen()
      * A private method for generating one frame as a JPEG
    - video\_feed()
      * The method connected to flask that allows for an HTTP request for one frame of video.
* Main flask application
  + Entry point for the flask user interface for the main web view
    - getCameraList
      * Method for retrieving the list of cameras.
    - getCameraListWithMotion
      * Gets a list of the cameras that actually have motion.
    - Index
      * The method connected to flask for serving up the main web view.
    - View\_camera
      * The method connected to flask for serving up the view of a single camera.
    - Home
      * The method connected to flask for serving up the home link.
    - Poll\_for\_status
      * Contains logic to return the activity logs and update the camera view.

5.3.2.3.5.2 Algorithmic model for each operation

* Video controller Program
* NN image recognition algorithm

It takes video frame as input and recognized human body on each frame, output coordinate and confident to draw box around body to draw box around face.

* OpenCV face object detection

It takes video frame as input and detect human face object on each frame, output coordinate.

* Person identification base on body feature.

Base on OpenCV face recognition, it catches upper body color as feature to identify differences amount people.

* Database

Not applicable

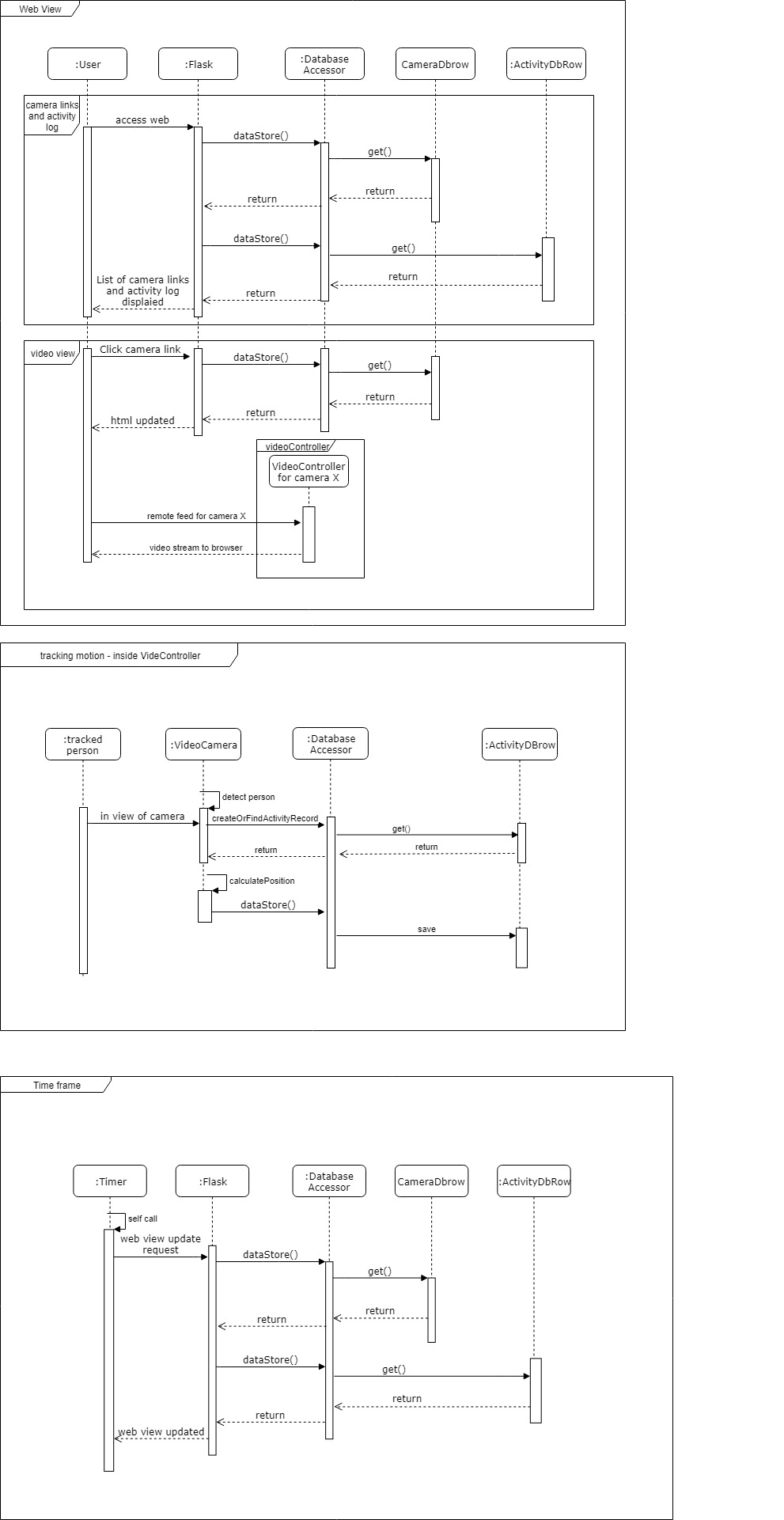
* Web View Program

Not applicable

* 1. Dynamic Behavior for Component n

Flask will generate the web main page and if a camera link is clicked, the VideoCamera will capture the cameras video feed and generates camera and activity related data. ActivityDbrow will store camera related data into the camera table and in the database. The CameraDbrow will store activity-related data into to the activity table. All information in the database will be used by Flask to update the web main page.

* + 1. Interaction Diagrams

**

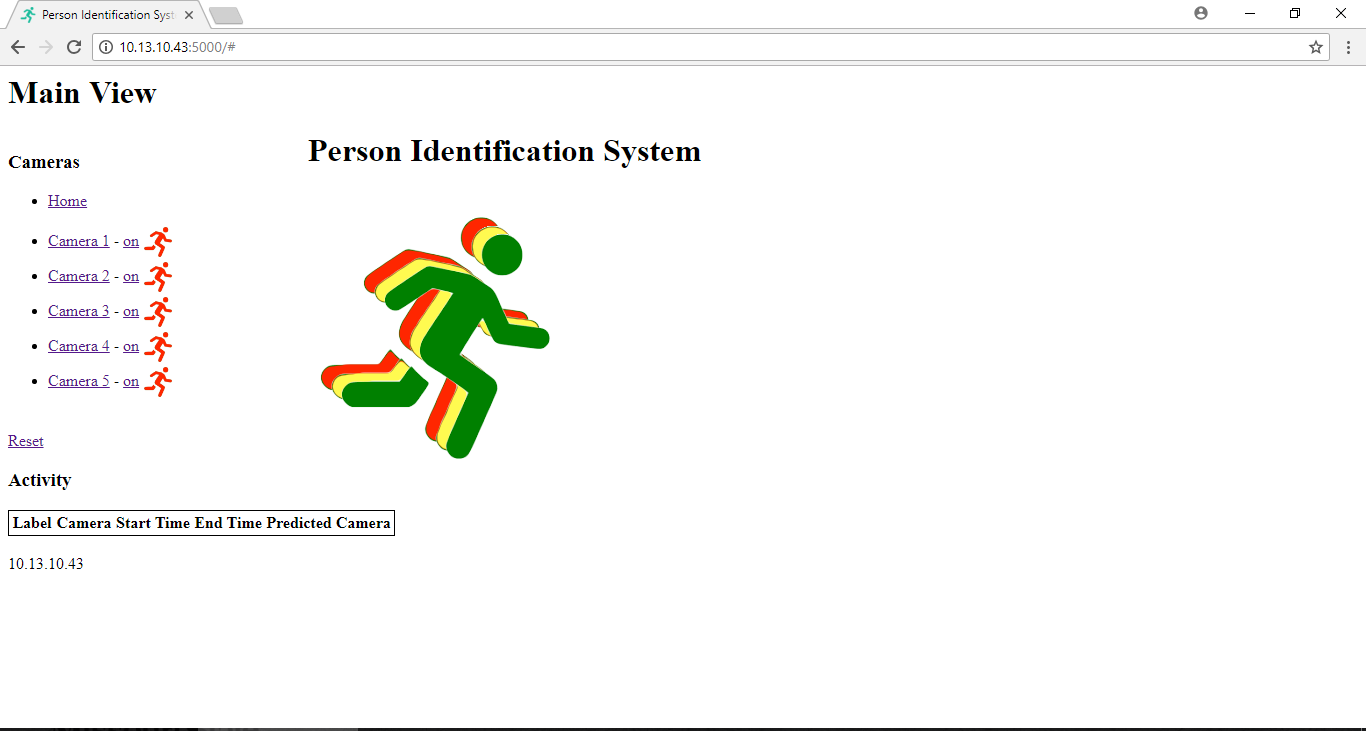
5.4.0 User interface design

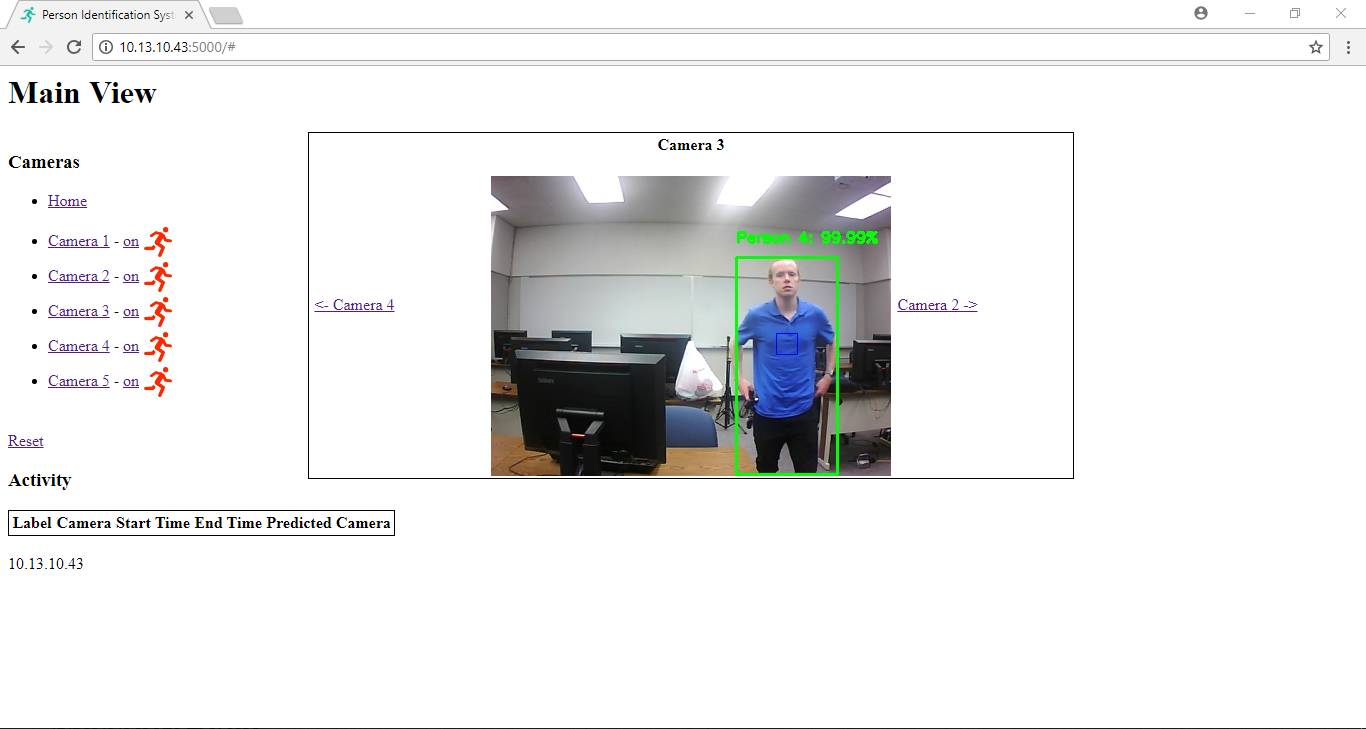
The user interface is how the user will interact with the Person Identification System. It will consist of a Home page, camera view links, and a textual log. Each of the pages will display useful information to the user and can be navigated by either a mouse or touch screen.

5.4.1 Description of the user interface

Each page will consist of standard GUI components, including but not limited to, buttons, text areas, and video anchors. These will be placed smartly on the interface so that even first time users will be able to quickly and efficiently navigate the user interface. These menus and their interactions with each other will be described in section 4.1.2

5.4.1.1 Screen images





5.4.1.2 Objects and actions

The following explains the Person Identification System interface:

* Home Page
  + The majority of this page will be encompassed by a help page that displays general information about the Person Identification System.
  + The textual movement log will be displayed along the bottom.
  + Clickable links will be located on the left side of the interface. These will allow you to navigate to the different camera views that have been connected to your network.
* Camera View
  + The Majority of this page will be encompassed by the selected video feed.
  + There will be clickable links to the left and right of the video feed that allow the user to quickly navigate to neighboring video feeds in the network.
  + On the left of the page clickable links to the Home page and the other camera views are available for further navigation.
  + The textual movement log is displayed on the bottom of the interface.

5.4.2 Interface design rules

* Navigation
  + The Camera links are on the left side of every page.
  + Camera view always has neighboring camera links on the left and ride of the video feed.
* Object Orientation
  + Camera feed is always displayed on the middle/right of the screen.
  + Textual log is always displayed on the bottom of the screen.

5.4.3 Components available

* Home button
  + Click it and go to the home view.
* Camera button
  + There is a clickable camera-view list that allows the user to view each camera’s videos.
  + On/Off button allows the user to turn the camera on and off.
* Textual log
  + Displays the activity log from the database.

5.4.4 UIDS description

* The basic structure of the User interface is created with HTML
* The basic design of the User interface is designed with CSS
* The functional aspects of the camera links and textual log will be implemented in JavaScript
* The basic web framework of the User interface is provided by Flask so, the Python application can communicate with a web-server

5.5.0 Restrictions, limitations, and constraints

The system requires a strong connection to the internet to work as the program retrieves data and the video is streamed over the internet. A good connection is required to retrieve clear, real-time video feeds from the cameras.

The system uses a network of ELP Mini USB cameras to record video feeds. Other cameras should not be mixed together because the algorithm for calculating the rectangle’s positions on the screen and other characteristics would be skewed if the camera’s video capturing qualities were different.

The system is based on a network of cameras that are connected to the main server. Each of these cameras must have a computer to run them, as well as an additional computer to run the server applications. The team is currently achieving this with a network of laptops that run the cameras, then these laptops are connected to one additional computer that runs the server application and database.

The computers need to have a 2.4 GHz Intel core i3 processor or higher. If the system is running on a processor with less power than this the performance will be decreased, and the system will not run at the optimum performance levels.

The implementation of cameras by the user could potentially increase or decrease the systems effectiveness. For example, if too few cameras are used in a building, or if the placement of cameras are not in good locations, the results would be reduced when recording a person’s location.

5.6.0 Testing Issues

Each section of the system was tested during development to ensure that it worked properly. When the program is completed it will be tested as a whole to guarantee that all components function together at expected.

5.6.0.1 White Box Testing

While each feature/class is being implemented, the team member working on that section will be responsible for testing and debugging that bit of code. If a problem cannot be worked out by one person, another team member may work on that section of code to help fix the issue. We will strive to ensure that each individual section of the code is working before being added to the system’s code.

5.6.0.2 Black Box Testing

After the system is assembled and working, we will have somebody who is unexperienced with computer technology to test our final version of the system.

5.6.0.3 Feature Testing

Each feature will have a test case associated with it, the following subsections will give a brief description of these test cases.

5.6.0.3.1 Access Main UI

* Description: Access web-based video hub.
  + **Input**: URL to access hub
  + **Output**: Video hub is successfully displayed

5.6.0.3.2 Video Select

* Description: Select a camera and its video feed will appear.
  + **Input**: Click camera link.
  + **Output**: Selected camera’s video feed is shown.

5.6.0.3.3 Motion Indicator

* Description: When a person walks into view, the motion indicator lights up next to the proper camera.
  + **Input**: Person walks into view of active camera
  + **Output**: Motion indicator lights up

5.6.0.3.4 Person Detection

* Description: When a person is in the camera view, they are framed and labeled.
  + **Input**: Person moves in into camera view
  + **Output**: Person is surrounded by labeled box

5.6.0.3.5 Destination Prediction

* Description: When a tracked person moves out of view toward another camera, the prediction indicator for that camera turns on.
  + **Input**: Tracked person leaves view of camera
  + **Output**: Prediction indicator appears on correct camera that the tracked person is heading towards.

5.6.0.3.6 Re-Identification

* Description: A previously tracked person will be re-identified when they come into view of a different camera.
  + **Input**: Tracked person leaves one camera and enters a different camera.
  + **Output**: Tracked person is identified as the same person and is reassigned the label they were given from the first camera.

5.6.0.3.7 Multiple Tracking

* Description: The system is able to track and re-identify multiple people at the same time.
  + **Input**: Person 1 and Person 2 each enter camera views, Person 1 and Person 2 then each move to a different camera.
  + **Output**: Person 1 and Person 2 both have correct motion and prediction indicators display on their respective cameras, also they are both correctly identified and labeled.

5.6.0.3.8 Textual Log View

* Description: Tracking logs are continuously printed in the activity panel as motion tracking occurs.
  + **Input**: Activity occurs on camera view
  + **Output**: Log is shown in Activity section of UI

5.6.0.3.9 Database connection

* Description: The application can successfully connect to the database on startup.
  + **Input**: Start the server application and cameras.
  + **Output**: The server application and cameras can communicate with the database.

5.6.0.3.10 On/Off Button Functionality

* Description: Camera feeds can be stopped/resumed by clicking the on/off button in the UI.
  + **Input**: On a connected camera, click the on/off button.
  + **Output**: The camera feed has stopped/resumed in accordance with the on/off click.

5.6.1 Performance bounds

Interactions between the client and the server should happen very quickly for optimum performance. This is because of how data is exchanged between the cameras and the server; a fast exchange rate is needed to receive real-time data about from the video feeds.

The server must maintain an acceptable level of performance to run the algorithms that make the system function. If the server computer is having performance issues and runs to slowly, the system’s performance will be impacted.

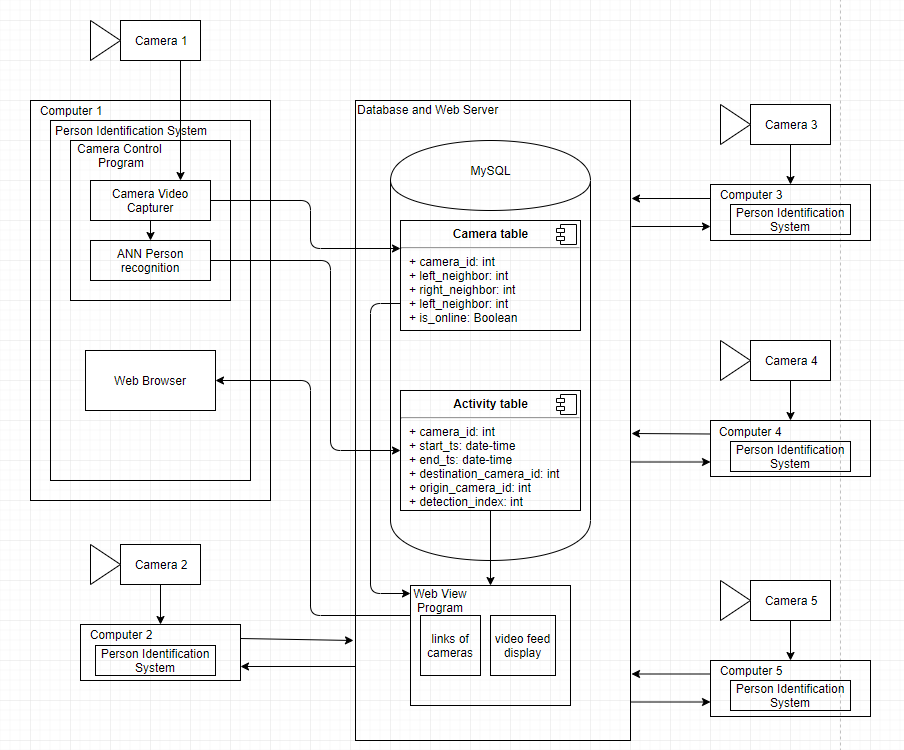
5.6.2 Identification of critical components

One of the critical components of the Person Identification System is the connection to the database. If this connection is compromised the program will not work. Camera-server interactions are also critical and required for the system to function. It is imperative that both connections work as expected.

The cameras themselves are another critical component. Since the system depends on receiving videos from these cameras, should they not function, the system will not work. The computers that the system is running on are also critical as they are required to keep the system functioning.

# Appendices

## Appendix I: Architectual Diagram



## Appendix II: CR Log Template

