SOFTWARE DESIGN DOCUMENT TEMPLATE (*modify as you deem appropriate*)

test

# 1.0 Introduction

The purpose of this software design document 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.

## 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.

## 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 ran 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.      Hardware scalability

3.      NN model upgrade

4.      Cameras to cloud database and web server connection with cloud or distributed computation on image processing.

**Input**:  Camera video feeds are sent to a cloud data server.

**Output**: Image processing occurs on the cloud and information is accessible to users of the cloud account anywhere.

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

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

## 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.

## 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.

# 2.0 Data design

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

## 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 method for shutting down the camera.
    - 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.

## 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)

);

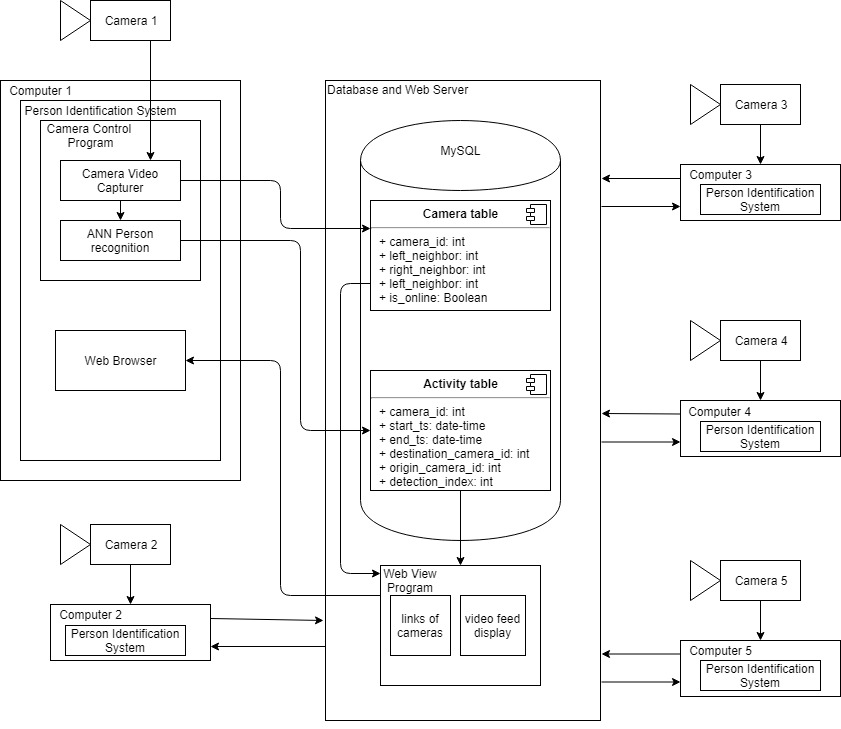
# 3.0 Architectural and component-level design

*A description of the program architecture is presented.*

## 3.1 System Structure

*A detailed description of the system structure chosen for the application is presented.*

### 3.1.1 Architecture diagram

**

## 3.2 Description for Component n (Your project may have multiple components)

*A detailed description of each software component contained within the architecture is presented. Section 3.2 is repeated for each of n components.*

### 3.2.1 Processing narrative for component n

*A processing narrative for component n is presented. It should describe the responsibilities of the component.*

### 3.2.2 Component n interface description.

*A detailed description of the input and output interfaces for the component is presented.*

### 3.2.3 Component n processing detail

*A detailed algorithmic description for each component is presented.*

#### 3.2.3.1 Design Class hierarchy for component n

#### 3.2.3.2 Restrictions/limitations for component n

#### 3.2.3.3 Performance issues for component n

#### 3.2.3.4 Design constraints for component n

#### 3.2.3.5 Processing detail for each operation of component n

##### 3.2.3.5.1 Processing narrative for each operation

##### 3.2.3.5.2 Algorithmic model for each operation

## 3.3 Dynamic Behavior for Component n

*A description of the interaction of the classes is presented.*

### 3.3.1 Interaction Diagrams

*A sequence diagram, for each use case the component realizes, is presented.*

# 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.

## 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

### 4.1.1 Screen images

### 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.

## 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.

## 4.3 Components available

* Home button
  + Click it and go to the home view
* Camera button
  + There are camera view lists links to allow us to get each camera videos
  + On/Off button allow us to work camera video or not
* Textual log
  + Displays the activity log from the camera views

## 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 function aspects of camera links and textual log will be implemented in JavaScript
* The basic web framework of the User interface is provided by flask. So, Python application can communicate with a web-server

# 5.0 Restrictions, limitations, and constraints

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

The system uses a network of ELP Mini USB cameras to record video feeds. Other cameras can be utilized on the program, however during the development process the ELP cameras were used primarily for testing and are currently the recommended device for the program.

The system is based on a network of cameras that connect 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 application. 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

The system that is running the server application needs to have a 2.4 GHz Intel core i3 processor or higher to properly run the algorithm. 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 the placement of cameras results in ineffective recording of people’s locations on a site, the overall effectiveness of the system could be reduced.

# 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 on whole to guarantee that all components function together at expected.

### 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.

### 6.0.2 Black Box Testing

After the system is assembled and working the team will being black box testing. This will help ensure that each feature of the system is functioning as expected. We will perform black box tests by testing all possible situations the program might see that are within our abilities.

### 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.

#### 6.0.3.1 Access Main UI

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

#### 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.

#### 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

#### 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

#### 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.

#### 6.0.3.6 Re-Identification

* Description: A previously tracked person will be re-identified when the 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.

#### 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.

#### 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

#### 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.

#### 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.

## 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.

## 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 interaction is 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 video 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.