THE UNIVERSITY OF THE WEST INDIES

ST. AUGUSTINE, TRINIDAD & TOBAGO, WEST INDIES

FACULTY OF ENGINEERING

**Department of Electrical & Computer Engineering**

BSc. in Electrical & Computer Engineering

**Software Design Document**

Covid Vision

Social Distance Counter using OpenCV



|  |  |
| --- | --- |
| Adeola Peters – 816000363 | Osahon Ojo – 816005001 |
| Adrian Williams – 816015427 | Lane Amichan – 816020580 |
| Avron Ramroop – 816021311 | Rashawn Thomas – 816018793 |
| Che Ferraz – 816021256 | Stephanie Babwah – 816008515 |
| Daniel Guiseppi – 816022933 | Verun Harbhajan – 816018987 |
| Nicholas Boodoo – 816007695 |  |

**Document Version Control**

**Project Name:** Social Distance Counter OpenCV

**Document Title:** Software Design Document

**Version:** 1.0.0a

**Date:** 14/12/2020

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| 14/10/2020 | 0.1.0d | Section 1 Introduction added, with some subsections deferred. |
| 15/10/2020 | 0.2.0d | Section 2 System Overview added. |
| 20/10/2020 | 0.2.1d | Section 1 Introduction updated to include an overview of that section.  Section 1.1 Purpose - updated wording.  Section 1.2 Scope updated to refer to current software version.  Section 1.4 Audience moved to Section 1.1 Purpose. |
| 09/11/2020 | 0.3.0d | Section 3 Data Design added. |
| 12/11/2020 | 0.3.1d | Section 4 Component Design added  References for OpenCV updated, references for COCO dataset and YOLO object detection added. |
| 12/11/2020 | 0.3.2d | Section 1.5 updated  Section 4.6 Graphical User Interface completed  Section 5 Human Interface Design added |
| 19/11/2020 | 0.4.0d | Section 3.2 updated for new system entities in GUI  Section 6 Requirement Matrix added |
| 4/12/2020 | 0.5.0d | Cover page and Table of Contents added  Section 1 minor changes in organization and wording  Section 3.2 updated for new system entities in GUI  Section 4 all algorithms converted to snippets  Section 5 images of GUI windows added |
| 13/12/2020 | 1.0.0a | Requirements matrix updated for graph |

Table of Contents

[1 Introduction 5](#_Toc57089716)

[1.1 Purpose 5](#_Toc57089717)

[1.2 Scope 6](#_Toc57089718)

[1.3 Overview 6](#_Toc57089719)

[1.4 Reference Material 7](#_Toc57089720)

[1.5 Definitions, Acronyms, and Abbreviations 7](#_Toc57089721)

[2 System Overview 8](#_Toc57089722)

[3 Data Design 9](#_Toc57089723)

[3.1 Data Description 9](#_Toc57089724)

[3.2 Data Dictionary 10](#_Toc57089725)

[4 Component Design 16](#_Toc57089726)

[4.1 Detection Network initialization 16](#_Toc57089727)

[4.2 Person Detection 17](#_Toc57089728)

[4.3 Image Annotation 18](#_Toc57089729)

[4.4 Distance Calculations 19](#_Toc57089730)

[4.5 Top View Transformation 19](#_Toc57089731)

[4.6 Graphical User Interface 19](#_Toc57089732)

[4.7 Account Verification 20](#_Toc57089733)

[5 Human Interface Design 20](#_Toc57089734)

[5.1 Login window 20](#_Toc57089735)

[5.2 Main window 22](#_Toc57089736)

[6 Requirements Matrix 24](#_Toc57089737)

# Introduction

This section is divided into five subsections to describe the purpose, scope, overview, and reference material for the project. A list of definitions, acronyms and abbreviations is also provided at the end of the section.

## Purpose

This Software Design Document (SDD) describes the considerations made for a social distance monitoring program using OpenCV. It outlines the decisions made in the design of the software, and the reasoning and/or constraints which led to them. This document provides an abstract view of the software, and describes its subsystems and how they interact with each other. Using this document, modifications to the design and implementation of the software can be made without changing the goals, conventions, and constraints of the current version. The current design can be analyzed to determine its suitability for a particular use case.

The primary audience includes programmers and managerial staff working on the Social Distance Counter project. The audience also includes reviewers of the design or source code and persons interested in the application of the software to their particular use case.

## Scope

This project focuses on the development and design of a Social Distance Counter using OpenCV. The primary objective is to create a product that monitors social distancing by detecting persons who violate the protocol. The software system monitors the number of persons within the live view from surveillance footage and determines whether each person is adhering to the social distancing protocol. It automatically computes the crowd density within a closed area and counts the number of violations to the protocol. This project can benefit businesses, educational institutions, workplaces, and other public places to ensure that proper social distancing measures are in place. The version of the Social Distance Counter accompanying this software design document is v 1.0.0a.

## Overview

This document describes how the software system will be structured for a Social Distance Counter to satisfy the requirements outlined in the System Requirements Specification (SRS). The system structure is described in five different sections to provide a detailed overview of the product. These sections include the system overview, data design, component design, human interface design and requirements matrix.

## Reference Material

1. OpenCV documentation

<https://www.docs.opencv.org/master/>

2. OpenCV Mat Datatype

https://docs.opencv.org/master/d6/d6d/tutorial\_mat\_the\_basic\_image\_container.html

3. COCO (Common Objects in Context) Dataset

<https://cocodataset.org/#home>

4. YOLO (You Only Look Once) Object Detection

https://pjreddie.com/darknet/yolo/

## Definitions, Acronyms, and Abbreviations

1. OpenCV: cross-platform image processing library
2. tkinter: Python library for making graphical user interfaces
3. SDD- Software Design Document
4. BGR- Blue Green Red

# System Overview

The system detects humans and places boundary boxes around each individual. The boxes are colour-coded to distinguish between the persons that follow and do not follow the social distancing protocol. A green box means that the protocol is followed, whereas a red box means that the persons are violating the protocol. The system automatically determines the crowd density (number of humans detected per pixel). The persons violating the protocol are automatically counted and the data is logged in a table format. The system uses the logged data to generate a graph showing the person count and violation count at different times. These functions are integrated to create a system that monitors social distancing.

This system was designed using the OpenCV package and implemented in python. It is useful due to the advent of the COVID-19 pandemic and crowd control must be monitored. It focuses on real time events, to track the response of persons to the recent social distancing protocol and ensure that proper social distancing measures are put in place.

# Data Design

This section begins the detailed design stage for the system. It focuses on how data structures were used to store and organize information and outlines a functional overview of the system. The section is divided into two sub-sections to discuss these points. These include the data description and data dictionary. The data description outlines a detailed explanation on the use of data structures in the system. The data dictionary tabulates all major system entities and describes the functions created for the completed product.

## Data Description

The social distancing monitoring program relies on image processing for identifying persons and determining the distances between each person detected. Images are the primary source of information input to the program. In the open source computer vision library, the primary data structure used is the Mat Object. This object is used to store images in terms of dimensions, channels, and depth. Images can be represented in either Blue Green Red (BGR) or Grayscale format. The BGR colour space was used to represent pixel values for this program. The Mat object represents each image as a matrix of pixel values. Videos are composed of a series of images hence the live surveillance footage is processed on a frame-by-frame basis by the software system. Each frame of the video is stored in the Mat datatype of OpenCV.

Lists were also used in functions to store and organize data. For example, the software system draws rectangles of different sizes around persons detected. The person detection algorithm uses lists to store coordinates for the boxes and the confidence levels. The contents of the lists change on a frame-by-frame basis for the persons detected. The person detection algorithm also requires the boxes to be scaled to the image width and height and these numerical operations are performed using Numpy arrays. The Numpy library provides useful tools to perform operations which can be combined with OpenCV. The Python bindings for OpenCV simplifies the use of data structures for storing and outputting images or video since the datatypes are neither explicitly declared nor manipulated but handled by the library.

A major entity is the pre-trained convoluted neural network used for object classification on each frame from the video input. The Common Objects in Context (COCO) dataset is stored as a text file containing a list of labels for the objects that can be identified. This social distancing program only requires the first label, “person”, but the convoluted neural network is trained on the entire dataset. A configuration file named “You Only Look Once” (YOLO) and YOLO weights file are the two other components required for object detection. These files provide the necessary data from the pre-trained convolutional neural network. The network can identify objects from the COCO dataset to varying confidence levels.

## Data Dictionary

The table below outlines the system entities in the GUI, along with their datatypes and a brief description.

| **Field Name** | **Data Type** | **Description** |
| --- | --- | --- |
| analyzeButton | tkinter.Button | Creates button “Show/Evaluate video file” to direct user to the feed selected with the social distance monitoring features implemented |
| appNameLabel | tkinter.Label | Labels app name: “COVID Vision” on the GUI |
| bodyFrame | tkinter.Frame | Frame holding all aspects of the GUI excluding the textbox |
| browseButton | tkinter.Button | Creates button “Browse” which directs user to their local disk (C:/Users/) to select a video file that have one of the extensions : “.avi”, “.mp4”, “.flv”, “.mkv”, “.mov”, “.wmv”. |
| buttonFrame | tkinter.Frame | Frame holding sign-in and create new account buttons |
| chooseVideoLabel | tkinter.Label | Labels instruction: “Choose a video file:” on main window |
| createButton | tkinter.Button | Create account button for the software system. Allows user to create a new account |
| displayButtonFrame | tkinter.Frame | Holds the buttons to display the logs (tabulated data) and graph |
| displayGraphButton | tkinter.Button | Creates button “Display graph” which directs user to graph: “Person Count and Violation Count vs Time” based on the selected video file and tabulated data |
| displayLogsButton | tkinter.Button | Creates button “Display log” which opens tabulated data with Time, Persons Detected, Crowd Density, and Violation Count |
| fileExplorerFrame | tkinter.Frame | Holds a label and button to browse the file explorer for a video file |
| filepathTextbox | tkinter.Text | Creates a textbox to the left of the browse button which stores the filepath for the selected video |
| formatLabel | tkinter.Label | Displays instructions for username and password creation/format |
| horScroll | tkinter.Scrollbar | Creates a horizontal scroll bar to shift left or right to view the reports |
| leftFrame | tkinter.Frame | Holds app icon label and app name label |
| loginWindow | tkinter.Toplevel | Login Window of GUI |
| logo | tkinter.PhotoImage | Stores an image (logo) for the GUI |
| logoLabel | tkinter.Label | Displays image (logo) on the left frame of the GUI |
| logoutButton | tkinter.Button | Creates button “Logout” which redirects user to the login window |
| mainLeftFrame | tkinter.Frame | Holds menu to view reports and camera feeds on the left frame of the main window |
| mainRightFrame | tkinter.Frame | Holds textbox to display reports on the right frame of the main window |
| mainWindow | tkinter.Tk | Main Window of GUI |
| menuLabel | tkinter.Label | Creates a label “Menu:” to distinguish the options to view reports or view the camera feeds |
| promptFrame | tkinter.Frame | Frame holding the textbox |
| quitInstructLabel | tkinter.Label | Labels instruction “Press ꞌꞌQꞌꞌ to exit video” below analyzeButton |
| quitInstructLabel2 | tkinter.Label | Labels instruction “Press ꞌꞌQꞌꞌ to exit video” below topViewButton |
| reportTextbox | tkinter.Text | Creates textbox which contains the vertical and horizontal scrolls and the tabulated data |
| rightFrame | tkinter.Frame | Holds prompt textbox, username and password fields, submit and create account buttons |
| submitButton | tkinter.Button | Sign-in button for the software system. Allows user to enter the system for live view of footage |
| textbox | tkinter.Text | Displays login screen prompt to user for basic instructions using the GUI |
| topViewButton | tkinter.Button | Creates button “Show top view” which opens the selected video file and outputs a top view of the video file |
| userFrame | tkinter.Frame | Frame holding username label and entry field |
| usernameEntry | tkinter.Entry | Entry box for user to enter their desired username |
| usernameLabel | tkinter.Label | Labels “Username:” on the GUI |
| vertScroll | tkinter.Scrollbar | Creates a vertical scroll bar to shift up or down to view the reports |

Several functions were created in the program to perform subtasks. These subtasks were then integrated to achieve the required functionality of the software system. The table below provides a brief description of each user-defined function.

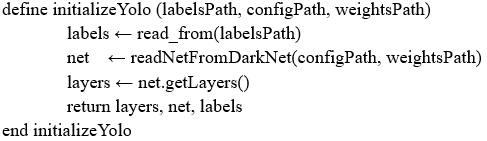
| **Field Name** | **Function Parameters** | **Return type** | **Description** | **Notes** |
| --- | --- | --- | --- | --- |
| annotateImage | image, indexes, boxes, confs, safeDistance | image- numpy.ndarray | Draws the boxes and prints person and violation count on each frame | Uses the highlight person function to draw the colour-coded boxes and norm to determine the distance between persons |
| highlightPerson | image, point, lengths, color, thickness, label, confidence | None | Used to draw boxes around persons | cv2.rectangle() method draws the boxes |
| initializeYolo | image, labelsPath, configPath, weightsPath | layerOutputs -list  labels- list | Opens model files and generates the layers | - |
| norm | x1, y1, x2, y2 | float | Used to compute distances | - |
| personDetect | image, layerOutputs, labels, confidenceLevel | indexes- numpy.ndarray  boxes- list  confs- list | Runs the detection on the layers from the image | confidenceLevel has a default value of 0.8 |
| runDetector | frame | frame- numpy.ndarray | All subtasks have been implemented in separate functions. This function combines subtasks to produce the required result on each frame | Uses initializeYolo, personDetect and annotateImage function calls |
| testDetector | None | None | Uses a sample video to verify the correct operation of the program | Multiple testDetector functions were created to test the program |

# Component Design

This section describes the components used to build the program. These components include modules for the detection of persons; calculations for the distances between persons, crowd density, and number of violations; annotations of the output image; transformation of the camera perspective to a top view; graphical user interface; and user account verification.

## Detection Network initialization

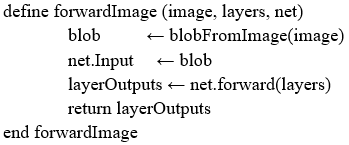
The OpenCV library is capable of loading a pre-trained object detection model. The YOLO object detection model was used for the program, and requires input files for the object labels, the YOLO network configuration, and the YOLO weights. Given an image, the model is used to create a set of “layers”, a data structure containing information about the objects detected in the image. The snippet below describes the algorithm for the “initializeYolo” function.



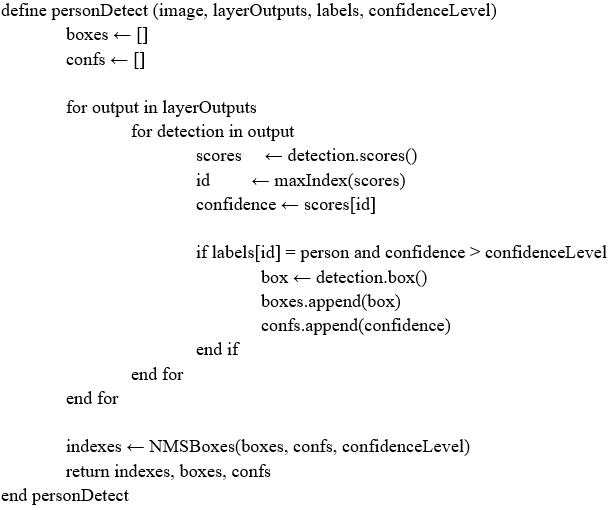
The initialization process involves a series of calls to functions from the OpenCV library to setup the network. The layers and network are obtained and returned. This function was made so that the initialization step is only run once at the start of the program. This function also isolates the rest of the program from the choice of using the YOLO network. If a different pre-trained model is to be used instead, the accompanying initialization process could replace the current one without major modification to the rest of the codebase.

## Person Detection

After obtaining the layers from the YOLO network, an image is read and forwarded to the network. The “forwardImage” procedure returns a set of “layerOutputs” that is then used for object detection. This procedure is described in the algorithm below.



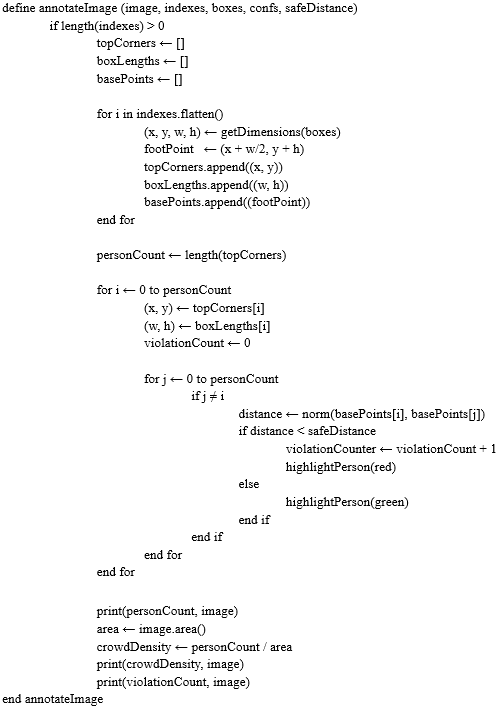
Each detection is stored in an output of the “layerOutputs”. The detections are iterated over, and if the associated label is that of a person, the position of the detection and the confidence of the detection is stored. Detections below a given confidence level are ignored. The “personDetect” algorithm runs the detection on the layers from the image as described in the snippet below.



The OpenCV library provides the method “NMSBoxes” for the removal of overlapping boxes. The “indexes”, “boxes”, and “confs” are returned and used to annotate the image. These boxes give the coordinates for drawing the bounding boxes around the persons detected on the image.

## Image Annotation

The “annotateImage” procedure uses the “safeDistance” parameter to determine if two persons are too close to each other. A green box is drawn around persons who are not too close to another person, and a red box is drawn around persons who are too close to another person. The algorithm describing the approach to this procedure is shown in the snippet below.



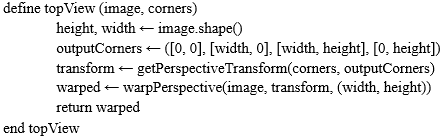
## Distance Calculations

The distance between two persons is calculated using the distance formula in 2D space as shown below.



## Top View Transformation

The OpenCV library provides the “getPerspectiveTransform” and “warpPerspective” methods used to obtain the top view of an image. The function requires 4 corners giving the (x, y) location of the pixels on the image that define the 4 corners of the region of interest on which the perspective transform is done. This process describes the “topView” function which is outlined step-by-step in the snippet below.



The outputCorners are a list of corresponding points on the output image where the input corners are mapped. The points were given clockwise from the top left corner.

## Graphical User Interface

The GUI was built using Python’s tkinter library. Section 5 of this document goes into further detail regarding the design of the GUI.

## Account Verification

The account verification component makes use of the graphical user interface; an accounts file, which contains all existing accounts; a database class, which accesses the accounts file to check for the existence of and create accounts; and an accounts class, which ensures the username is valid and uses the database class to determine whether or not the user can log in or create an account. When the user attempts to log in, the accounts class first checks that the account to be logged into exists and when the user attempts to create an account, the accounts class first checks that the account does not already exist.

# Human Interface Design

The system uses a GUI since it is more user-friendly than a command-line interface. The GUI is split into two main windows: the login window and the main window.

## Login window

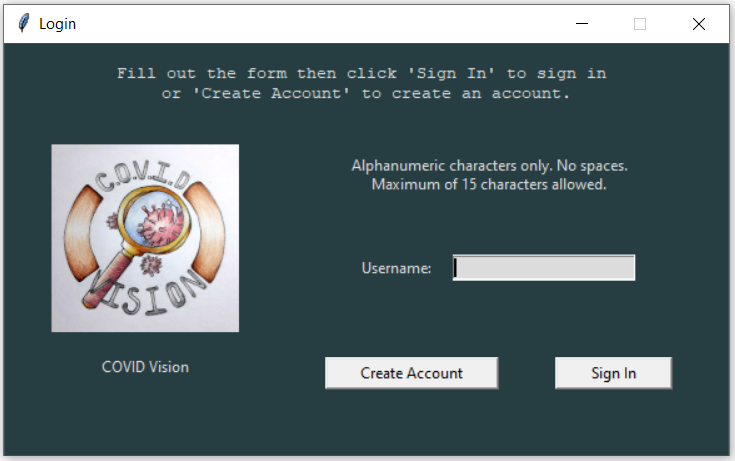
On startup, the user sees the login window. This window allows the user to either log in to his or her account or create a new account. After either action (logging in or creating an account), the user is granted access to the main window. The login window is visually divided into three sections: a textbox and two frames.

The first and topmost section of the login window is a textbox that spans the entire width of the window. This textbox instructs the user about how to use the login window, that is, how to log in and how to create a new account. This textbox also alerts the user if he or she enters invalid data when logging in or tries to create an account under a username that already exists.

The next two sections are frames that lie directly beneath the textbox and in-line with each other. Together, they span the width of the login window. The frame on the left contains two elements. The topmost of these is a label that displays the application’s icon. There is another label directly beneath this, which displays the application’s name.

The frame on the right contains several elements. The topmost element is a label that displays three criteria for a valid username. Directly beneath this label, there is a frame that holds the entry field into which the user will type his or her username. To the left of and in-line with this entry field, there is a label which indicates that the entry field accepts a username. Beneath the username frame is another frame that contains two buttons placed in-line with each other. The button on the left allows the user to create a new account and the button on the right allows the user to log into the application.

The figure below illustrates the login window of the GUI.

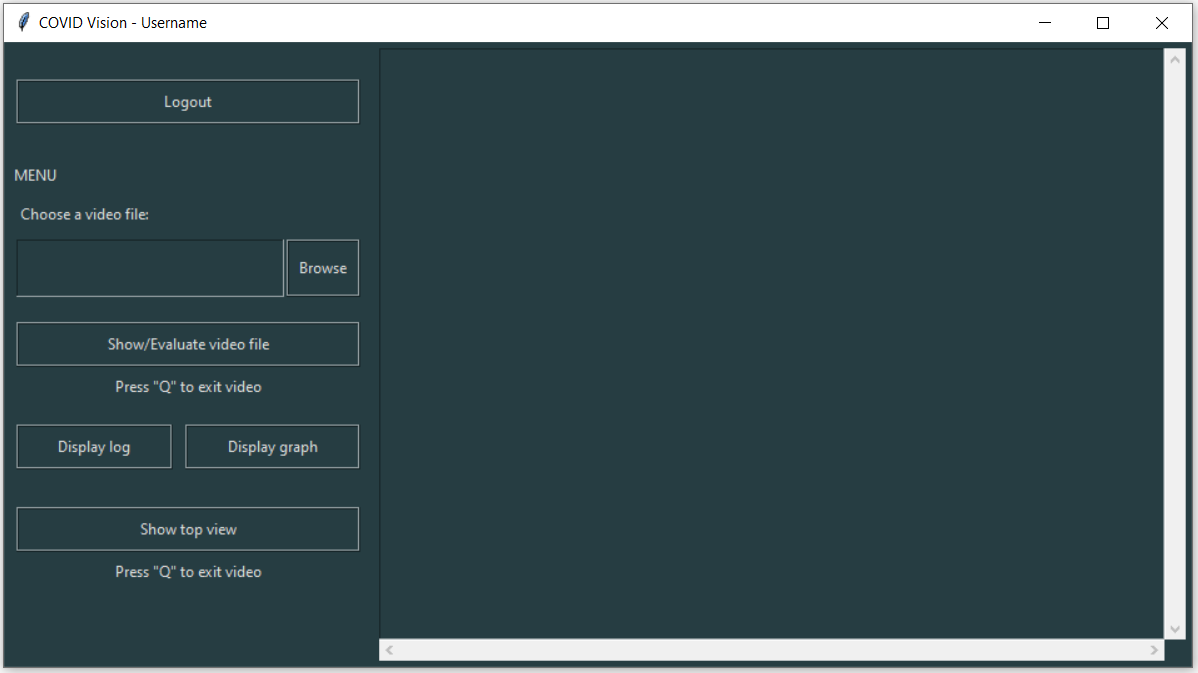


## Main window

The main window allows the user to browse for a video file, analyze the video file using OpenCV, generate logs from the analysis and display those logs together with a graph. It also allows the user to return to the login window. The main window is visually divided into two frames, placed in-line with each other.

The frame on the left contains several elements and has a set width. The topmost is a label that, when clicked on, returns the user to the login window. Directly beneath this, there is a label indicating that the succeeding elements comprise a menu. The first element in the menu is a label that instructs the user to choose a video file to be analyzed. The next element in the menu is a frame that contains a label and a button in-line with each other. Clicking the button opens a file explorer window that allows the user to choose a video file. After the user chooses a video file, the label displays the file’s path. Beneath the file explorer frame, there is a button to analyze the chosen video file. There is another frame containing two buttons in-line with each other below the analysis button. The left button displays the logs generated from the video file analysis. The right button generates a graph based on the logged data, using the python library “Matplotlib”.

The frame on the right contains one element: a textbox that spans most of the window’s width. When the generate logs button is clicked, said logs are displayed in the textbox. The graph is generated as a figure in a separate window. Below the logs and graph button, there is one more button to show the top view of the selected video file. The figure below illustrates the main window of the GUI.



# Requirements Matrix

The table below provides a cross-reference to illustrate the system components which satisfied each of the functional requirements from the SRS.

| **SRS Requirements** | | | | **System Design** | | **System Defects** |
| --- | --- | --- | --- | --- | --- | --- |
| **Label** | **Product Requirements** | **Priority** | **System Specifications** | | **Requirement Achieved** |  |
| REQ1-1 | The system shall detect humans only | Necessary | The personDetect algorithm runs the human detection on the layers it receives from the YOLO network | | Yes | Only humans are detected but the pre-trained network can fail to recognize a person due to confidence levels |
| REQ1-2 | The system shall place coloured boundary boxes based on whether the social distancing protocol was followed | Necessary | The highlightPerson algorithm draws the coloured boundary boxes | | Yes | No defects |
| REQ1-2.1 | The boundary boxes shall be red if someone is in violation of the social distancing protocol. | Necessary | The annotateImage algorithm uses tbe highlightPerson algorithm to draw red boundary boxes if the distance between persons is less than the safe distance | | Yes | Colour-switching of boxes is delayed at times if persons are not detected or if a group of persons is interpreted as an individual |
| REQ1-2.2 | The boundary boxes shall be green if someone is adhering to the social distancing protocol | Necessary | The annotateImage algorithm uses tbe highlightPerson algorithm to draw green boundary boxes if the distance between persons is greater than or equal to the safe distance | | Yes | Colour-switching of boxes is delayed at times if persons are not detected or if a group of persons is interpreted as an individual |
| REQ1-2.3 | The boundary boxes shall cover the entire human | Necessary | The personDetect algorithm returns the indexes which are used to obtain the dimensions for the boxes in the annotateImage algorithm | | Yes | No defects |
| REQ1-3 | The system shall count the number of persons in the room of specified dimension | Necessary | Using the boxes and indexes returned from the personDetect algorithm, annotateImage determines the personCount from the boxes | | Yes | No defects to the algorithm but the result depends on if all persons are detected |
| REQ1-3.1 | The system shall count the number of persons in violation of the social distancing protocol in the specified space | Necessary | The annotateImage algorithm initializes violationCounter to 0 and increments if the distance between persons are less than the safe distance (this counts as 1 violation) | | Yes | No defects |
| REQ1-3.2 | The system shall take into consideration time for persons to pass each other in walkways before indicating a breach in protocol. (approximately 30s) | Highly Desirable | The system does achieve this specification. This feature can be added to improve the system design | | No | Requirement was not achieved |
| REQ1-3.3 | The system should automatically log the crowd density, social distance metric and the number of persons violating the social distancing protocol and the duration of the violation in real time | Necessary | The annotateImage algorithm automatically computes the crowd density and a report is generated with all logged data | | Partially | The system does not log the duration of the violation. Instead, it logs the time that violation occurs. |
| REQ1-3.4 | The system shall visually display the information from the reports in the form of graphs | Desirable | The GUI has a viewGraph function which uses matplotlib to generate a graph of “Person Count and Violation Count vs. Time” | | Yes | No defects |
| REQ2-1 | The system will be able to operate on the following two OS: MAC, Windows | Highly Desirable | The system operates on the MAC, Linux and Windows operating system | | Yes | No defects |