Bilkent University

Department of Computer Engineering

Senior Design Project

Third Eye

Low-Level Design Report

Alkım Önen, İbrahim Eren Tilla, Göktuğ Öztürkcan, Berke Oğuz, Safa Alperen Oruç

Supervisor: Fazlı Can

Jury Members: Can Alkan and Çiğdem Gündüz Demir

Low-Level Design Report

Feb 7, 2021

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Contents

1. Introduction

This is the Low-Level Design Report of our project, *Third Eye*. In this report, we will discuss the object design trade-offs, interface documentation guidelines, engineering standards, packages and class interfaces.

A year ago, most of us did not know about a term which we are using in everyday life nowadays: social distancing. With the emergence of the COVID-19, all our lives had changed, and we had to adapt accordingly [1]. Although the virus itself affected the whole world immensely, both in an economic and in a social sense, the world keeps spinning and people keep on living with their lives [2]. With it comes the need for social distancing.

Social distancing, in its most basic form, is to refrain from getting close with other people in crowded spaces. Research about social distancing proved it to be useful in ceasing the spread of the virus to a crucial point. It is believed that 3-4 months of moderate social distancing can save up to 1.7 million lives and $8 trillion or $60,000 per US household [3]. Thus, it is of utmost importance to realize the role of social distancing in fighting COVID-19.

Although the benefits of social distancing are undoubtable, there are still many people that do not care about it and keep on violating spacing rules and advice in crowded spaces [4]. This brings the need for a regulation system for the crowds and that is where *Third Eye* steps in. With our project, we aim to put this problem on the table and come up with a viable solution that will minimize this issue.

However, *Third Eye* does not only aim to regulate social distancing but to provide useful data to the users as well. It will analyze the crowds and where the people's interest peaks. Then it will inform the users how the crowd moves, and which parts of the market are trending. For example, a shopping center overseer will be able to check out the data that *Third Eye* offers and see the trending sections of the mall.

1.1 Object design trade-offs

1.1.1 Functionality vs Usability

Functionality and usability are the most important aspects while considering our projects design trade-offs. Since our users will be shopping malls security department, we offer a self-explanatory and user-friendly project. Our project is also clear in terms of display and language and presents a neat and well-organized user interface. However our project, *Third Eye*, offers excellent functionalities, we can say that our design favors usability over functionality.

1.1.2 Efficiency vs Accuracy

Third Eye will have the feature to work and process both live camera feed and already recorded video footages. The system should work in real time and should not have delays with its results while in live mode. Similarly, it should not keep waiting users for extended amounts of time in prerecorded mode. On the other hand, the human detection algorithms working in the system should be accurate. Otherwise, some other objects may be confused as humans and the system may not work as it is supposed to. Therefore, while both of these aspects are important for us, our design leans a little bit towards efficiency.

1.1.3 Extensibility vs Robustness

Robustness is indeed important however we are making a compromise since the Covid19 pandemic will be over at some point which *Third Eye* will need to be extended to another use case. In that regard, we are tipping to the extensibility of the application while diminishing robustness.

1.1.4 Reliability vs Cost

Third Eye aims to provide a more reliable service. The system runs and collects data 24/7 and makes sure that user data is safely stored. *Third Eye* also makes sure that the data produced is correct. Obviously, creating a reliable system is costly in terms of money and time. Even though reliability of *Third Eye* is costly, we prefer our system to be more reliable so our design favors reliability over cost.

1.2 Interface documentation guidelines

class `Sample`	
This class is responsible for...	
Attributes	
`public String name` `private int id`	
Methods	
`public String getName()`	This method returns the name.
`public int verify(id)`	This method verifies the ID.

Table 1: Class Interface Outline

1.3 Engineering standards

UML is a standardized modeling language in the field of software engineering. The report holds to the UML guidelines in terms of subsystem composition, use cases, scenarios, class interfaces and diagrams. IEEE develops global standards in a broad range of industries. The report follows IEEE's standards for the citations.

1.4 Definitions, acronyms, and abbreviations

UML: Unified Modeling Language. It is a standard language for specifying, visualizing, constructing and documenting the artifacts of a software system.

IEEE: Institute of Electrical and Electronics Engineers. It is an association that develops global standards in a broad range of industries.

GUI: Graphical User Interface. It is a common user interface that includes graphical representations.

UI : User Interface. The user interface (UI) is the point of human-computer interaction and communication in a device.

AI: Artificial intelligence refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions.

2. Packages

Third Eye's low level system is composed of 3 different layers. These are the presentation, application and data layer. All of these parts are located in the client side since *Third Eye* is an offline application and does not use a server.

2.1 Presentation Tier

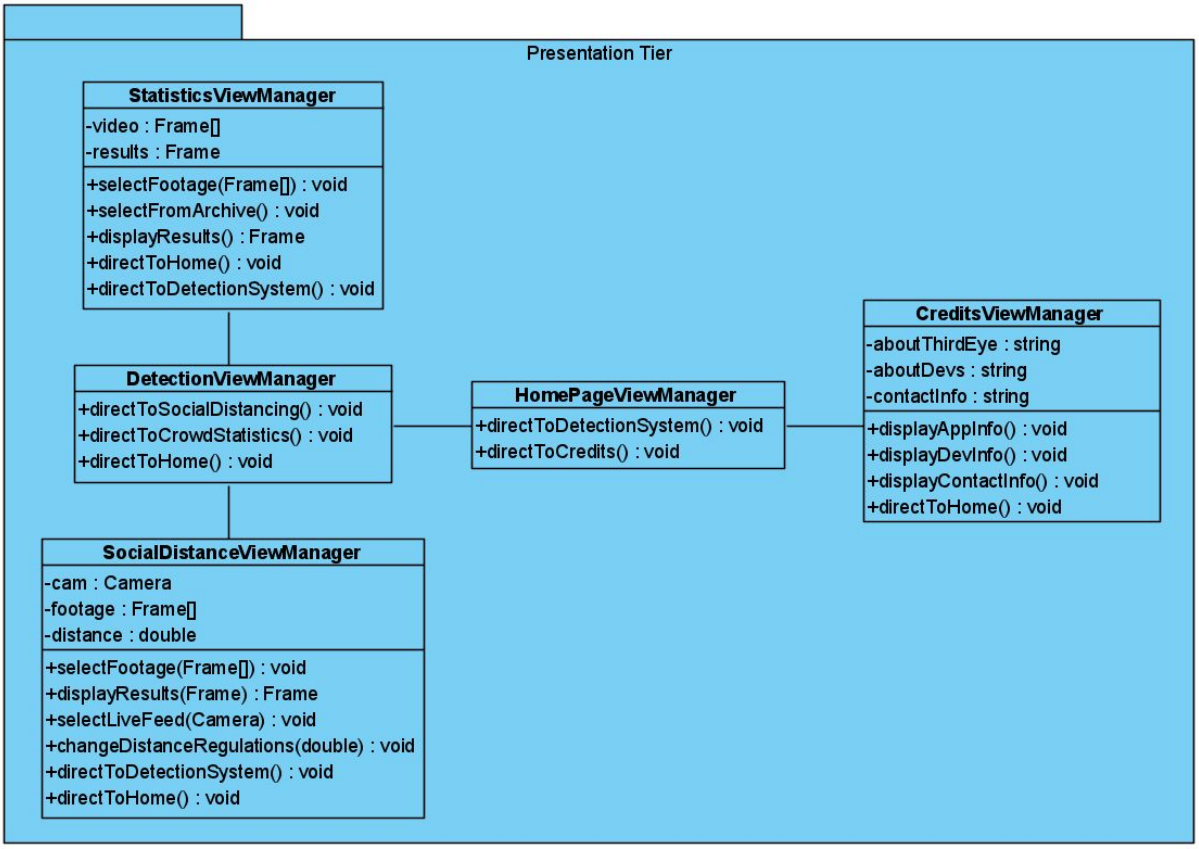

Figure 1: Subsystem Decomposition for Presentation Tier

This subsystem will be the connection between the users and the application tier itself. It will control the composition and navigation of the user interfaces. This subsystem will have GUI implementations with practical functionalities and page designs. Every page will have a class to operate its functions. The presentation tier will be connected to the Application Tier in order to perform the actions desired by the client.

HomePageViewManager

This class is responsible for welcoming users and initiating the detection system on command.

DetectionViewManager

This class is responsible for selecting the detection modes.

SocialDistanceViewManager

This class is responsible for showing social distancing violations in given footage or live feed.

StatisticsViewManager

This class is responsible for showing crowd statistics of a selected footage.

CreditsViewManager

This class is responsible for showing general information about *Third Eye*, developers and their contact information.

2.2 Application Tier

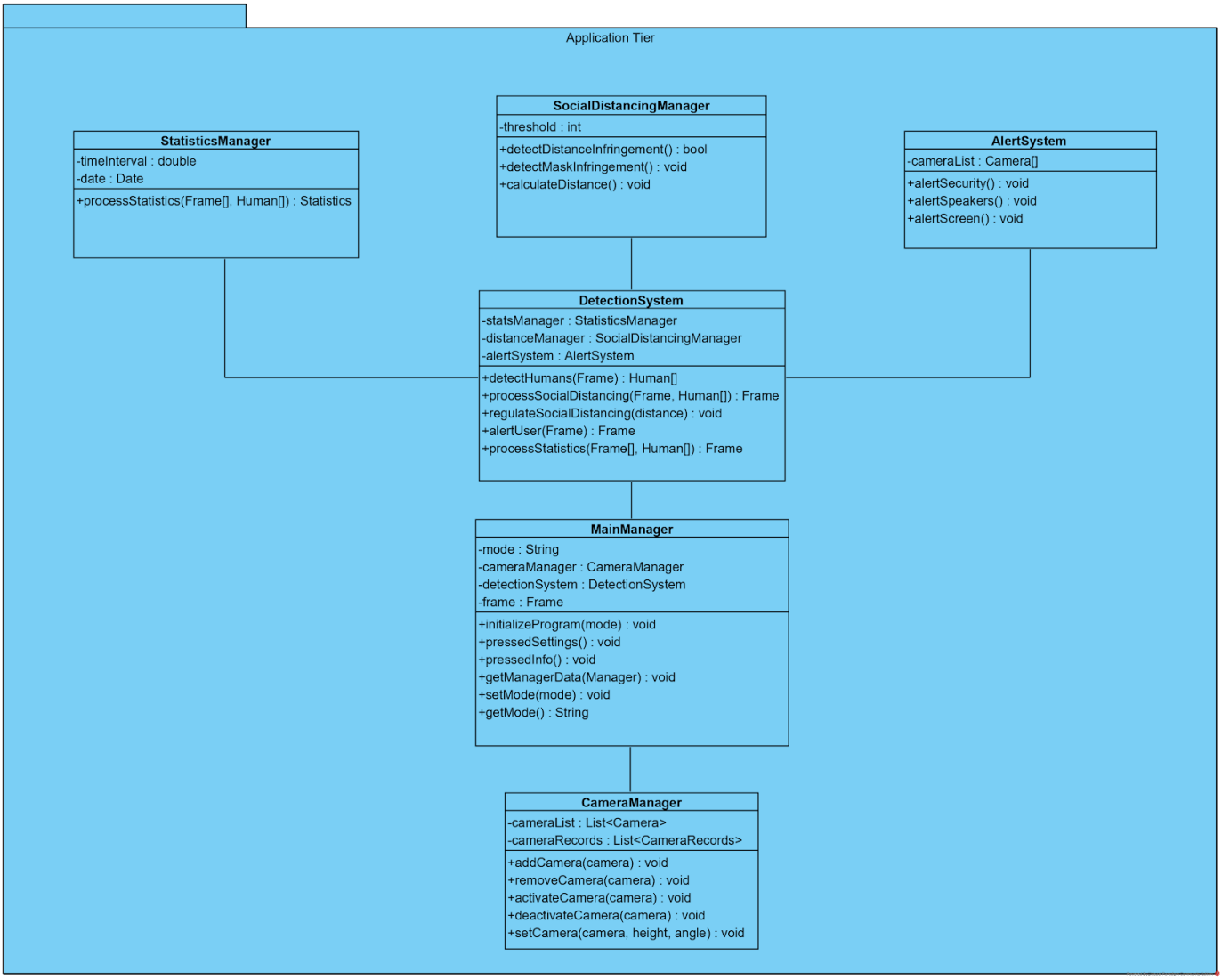

Figure 2: Subsystem Decomposition for Application Tier

All the manager interactions are done in this layer. This layer helps the parts of the application (such as camera, alert system and main manager) work together in sync.

MainManager

MainManager is responsible for the management of all the other manager classes. It is the backbone of *Third Eye* in that regard. It holds other managers as instances and works as a bridge and a constructor between the said classes. MainManager can be said to be the "main" class of our application.

CameraManager

CameraManager is responsible for the management of camera feed as well as the cameras. It's main capabilities are to add, remove, set or list cameras.

DetectionSystem

This class is responsible for human detection on frames coming from the camera and video footages.

SocialDistancingManager

SocialDistancingManager is responsible for the management of the social distancing function of ThirdEye. It does the most important function of ThirdEye which is enforcing social distancing rules.

AlertSystem

AlertSystem works with the other classes and is used to give feedback to the user in case any of the social distancing rules are broken.

StatisticsManager

This class is responsible for creating the crowd statistics in given video footage.

2.3 Data Tier

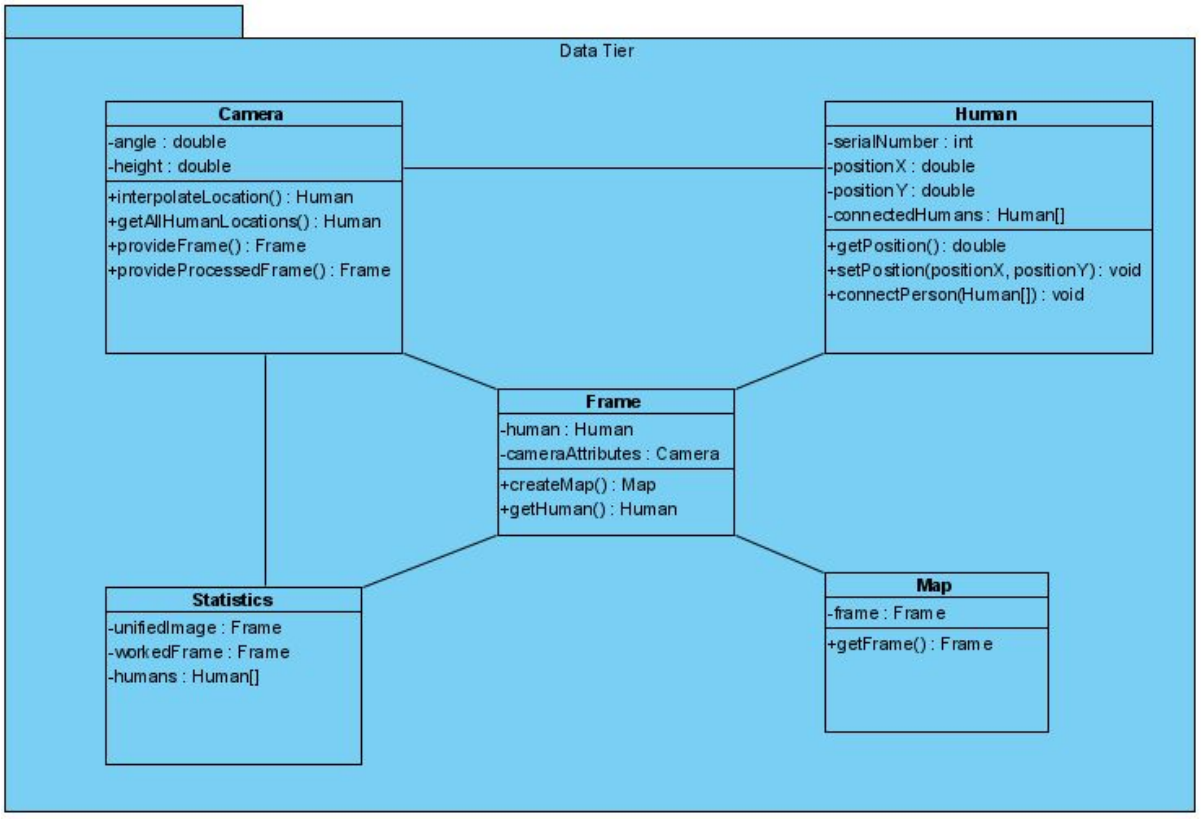

Figure 3: Subsystem Decomposition for Data Tier

Data Tier is responsible for the storage of the images and statistics. *Third Eye* does not record the camera feeds in any database for privacy concerns. The main reason for using the data tier is to take the local data to process the camera feeds.

Camera

This class is responsible for keeping information about the camera that is providing the video footage.

Human

A class for human objects on camera. This class is used to define humans with their serial numbers which are assigned by the program, and also group them if they are relatives.

Statistics

This class is responsible for storing where the people gather in the location of the image.

Frame

A class for creating map and data for processing according to camera feed.

Map

A class for 2D map object

3. Class Interfaces

3.1 Presentation Tier

class `HomePageViewManager`	
This class is responsible for welcoming users and initiating the detection system on command.	
Methods	
`public void directToDetectionSystem()`	Directs application to main detection view.
`Public void directToCredits()`	Directs application to credits view.

Table 2: HomePageViewManager Interface

class `DetectionViewManager`	
This class is responsible for selecting the detection modes.	
Methods	
`public void directToSocialDistancing()`	Directs application to social distancing view.
`public void directToCrowdStatistics()`	Directs application to crowd statistics view.
`public void directToHome()`	Directs application to home page view.

Table 3: DetectionViewManager Interface

class `SocialDistanceViewManager`	
This class is responsible for showing social distancing violations in given footage or live feed.	
Attributes	

`private Camera cam`
`private Frame[] footage`
`protected double distance`	

Methods	
`private void selectFootage(Frame[])`	Finds the video footage to be processed.
`public Frame displayResults(Frame)`	Processes social distancing and show results.
`private void selectLiveFeed(Camera)`	Selects the camera to get the live feed.
`public void changeDistanceRegulations(distance)`	Changes the social distancing regulations.
`public void directToDetectionSystem()`	Directs application to main detection view.
`public void directToHome()`	Directs application to home page view.

Table 4: SocialDistanceViewManager Interface

class `StatisticsViewManager`	
This class is responsible for showing crowd statistics of a selected footage.	
Attributes	
`private Frame[] video` `protected Frame results`	
Methods	
`private void selectFootage(Frame[])`	Finds the footage to be processed.
`private void selectFromArchive()`	Selects statistics from processed archives.
`public Frame displayResults()`	Processes crowd statistics of given footage.
`public void directToHome()`	Directs application to home page view.
`public void directToDetectionSystem()`	Directs application to main detection view.

Table 5: StatisticsViewManager Interface

class `CreditsViewManager`	
This class is responsible for showing general information about *Third Eye*, developers and their contact information.	
Attributes	

`private string aboutThirdEye`
`private string aboutDevs`
`private string contactInfo`	

Methods	
`public void displayAppInfo()`	Displays about *Third Eye* section.
`public void displayDevInfo()`	Displays about developers section.
`public void displayContactInfo()`	Displays contact information section.
`public void directToHome()`	Directs application to home page view.

Table 6: CreditsViewManager Interface

3.2 Application Tier

class `MainManager`	
MainManager is responsible for the management of all the other manager classes. It is the backbone of the *Third Eye* in that regard. It holds other managers as instances and works as a bridge and a constructor between the said classes. MainManager can be said to be the "main" class of our application.	
Attributes	

`protected string mode`
`protected CameraManager cameraManager`
`protected DetectionSystem detectionSystem`
`protected Frame frame`	

Methods	
`public void initializeProgram(mode)`	Initializes *ThirdEye* in the regarding mode.
`public void pressedSettings()`	Directs to Settings page.
`public void pressedInfo()`	Directs to Info page.
`public void getManagerData(Manager)`	Retrieves the data of the regarding Manager.
`public void setMode(mode)`	Sets the mode of the application.
`public string getMode()`	Returns the mode of the application.

Table 7: MainManager Interface

class `CameraManager`	
CameraManager is responsible for the management of camera feed as well as the cameras. It's main capabilities are to add, remove, set or list cameras.	
Attributes	
`protected List<Camera> cameraList` `protected List<CameraRecords> cameraRecords`	
Methods	
`public void addCamera(camera)`	Adds a camera to the system.
`public void removeCamera(camera)`	Removes a camera to the system.
`public void activateCamera(camera)`	Activates an already registered camera.
`public void deactivateCamera(camera)`	Deactivates an already registered camera.
`public void setCamera(camera, height, angle)`	Sets the height and angle of the specified camera.

Table 8: CameraManager Interface

class `DetectionSystem`	
This class is responsible for human detection on frames coming from the camera and video footages.	
Attributes	

`protected StatisticsManager statsManager`
`protected SocialDistancingManager distanceManager`
`protected AlertSystem alertSystem`	

Methods	
`Public Human[] detectHumans(Frame)`	Detects the humans in the given frame.
`public Frame processSocialDistancing(Frame, Human[])`	Processes the image to check violations of social distance regulations.
`public void regulateSocialDistancing(distance)`	Changes social distance measures with given numbers.
`public Frame alertUser(Frame)`	Alerts the user and provides the violation of social distance regulations in an image.
`public Frame processStatistics(Frame[], Human[])`	Processes given frames in order to form crowd statistics of footage.

Table 9: DetectionSystemInterface

class `SocialDistancingManager`	
SocialDistancingManager is responsible for the management of the social distancing function of ThirdEye. It does the most important function of ThirdEye which is enforcing social distancing rules.	
Attributes	
`protected int threshold`	
Methods	
`public bool detectDistanceInfringement()`	Catches if social distancing threshold rule is broken.
`public void detectMaskInfringement()`	Checks if wearing a mask rule is broken.
`public void calculateDistance()`	Calculates distance between the humans in the frame.

Table 10: SocialDistancingManager

class `AlertSystem`	
AlertSystem works with the other classes and is used to give feedback to the user in case any of the social distancing rules are broken.	
Attributes	
`protected Camera cameralist[]`	
Methods	
`public void alertSecurity()`	Gives a specific alert to the security that social distancing rules are broken.
`public void alertSpeakers()`	Automatically gives an alert sound from the speakers when social distancing rules are broken.
`public void alertScreen()`	Automatically gives an alert sound from the speakers when social distancing rules are broken.

Table 11: AlertSystem Interface

class `StatisticsManager`	
This class is responsible for creating the crowd statistics in given video footage.	
Attributes	
`protected double timeInterval` `protected Date date`	
Methods	
`public Statistics processStatistics(Frame[], Human[])`	Processes given frames in order to form crowd statistics of footage.

Table 12: StatisticsManager Interface

3.3 Data Tier

class `Frame`	
A class for creating map and data for processing according to camera feed.	
Attributes	
`private Human human` `private Camera cameraAttributes`	
Methods	
`public Map createMap()`	Creates a bird's eye map by processing frames
`public Human getHuman()`	Returns human objects on camera frame

Table 13: Frame Interface

class `Map`	
A class for 2D map object	
Attributes	
`private Frame frame`	
Methods	
`public Frame getFrame()`	This method gets the camera frame.

Table 14: Map Interface

class `Human`	
A class for human objects on camera. This class is used to define humans with their serial numbers which are assigned by the program, and also group them if they are relatives.	
Attributes	

`private int serialNumber`
`private double positionX`
`private double positionY`
`private Human[] connectedHumans`	

Methodsn	

`public int getPosition()`
 	
`public void setPosition(double x , double y)`
 	
`public void connectPerson(Human[] connectedHumans)`
	This method returns the position of humans as coordinates.
	
	This method assigns position to human as x-y coordinates.
	
	Connect people if they are relative. It is important in order to neglect the social distance rules between those people.

Table 15: Map Interface

class `Camera`	
This class is responsible for keeping information about the camera that is providing the video footage.	
Attributes	
`protected double angle` `protected double height`	
Methods	
`private Human interpolateLocation(Human)`	Interpolates provide the object's location into the real world plane.
`public Human[] getAllHumanLocations(Human[], Frame)`	Interpolates the locations of all humans detected in the frame.
`public Frame provideFrame()`	Returns the current frame from a live feed.
`public Frame provideProcessedFrame()`	Returns the current frame after human detection.

Table 16: Camera Interface

class `Statistics`
This class is responsible for storing where the people gather in the location of the image.
Attributes

`protected Frame unifiedImage`
`protected Frame workedFrame`
`protected Human[] humans`

Table 17: Statistics Interface

4. Glossary

GitHub: An online platform that provides hosting for software development version control using Git.

Jira: A software that helps manage agile and software development projects.

OpenCV: A library of programming functions mainly aimed at real-time computer vision developed by Intel[9].

Jetson Nano: A development kit for AI applications that reduces complexity and effort for developers. Jetson Nano is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All in an easy-to-use platform that runs in as little as 5 watts [10].

Computer Vision: Computer vision is a field of artificial intelligence that trains computers to interpret and understand the visual world.

Tkinter: It is a standard Python interface to the Tk GUI toolkit.

5. References

[1] "Naming the coronavirus disease (COVID-19) and the virus that causes it," World Health Organization. [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/technical-guidance/naming-the-coronavirus-disease-(covid-2019)-and-the-virus-that-causes-it. [Accessed: 11-Oct-2020].

[2] I. Chakraborty and P. Maity, "COVID-19 outbreak: Migration, effects on society, global environment and prevention," Science of The Total Environment, vol. 728, p. 138882, 2020.

[3] M. Greenstone and V. Nigam, "Does Social Distancing Matter?," SSRN Electronic Journal, 2020.

[4] K. Kelland and M. Revell, "Pandemic behavior: Why some people don't play by the rules," The Jakarta Post, London, 13-Aug-2020.

[5] S. Cole, Security Cameras Are Keeping Track of Social Distancing in Public Spaces, 13-Apr-2020. [Online]. Available: https://www.vice.com/en/article/9394n7/live-cam-social-distancing-trends-in-real-time. [Accessed: 11-Oct-2020].

[6] "Behavior Analytics," Axxonsoft. [Online]. Available: https://www.axxonsoft.com/intelligence/behavioranalytics.php. [Accessed: 11-Oct-2020].

[7] Paul, Manoranjan & Haque, Shah & Chakraborty, Subrata. (2013). "Human detection in surveillance videos and its applications - A review. EURASIP Journal on Advances in Signal Processing." 2013. 25. 10.1186/1687-6180-2013-176.

[8] "Code of Ethics," Code of Ethics | National Society of Professional Engineers. [Online]. Available: https://www.nspe.org/resources/ethics/code-ethics. [Accessed: 10-Oct-2020].

[9] "OpenCV" , About OpenCV [Online]. Available: https://opencv.org/about/ [Accessed: 21-Nov-2020].

[10] " NVIDIA Jetson Nano" , Jeton Nano [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit#:~:text=NVIDIA%C2%AE%20Jetson%20Nano%E2%84%A2,as%20little%20as%205%20watts. [Accessed: 21-Nov-2020].

[11] A. Rosebrock, "Histogram of Oriented Gradients and Object Detection," PyImageSearch, 10-Nov-2014. [Online]. Available: https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/. [Accessed: 21-Oct-2020].

[12] "Complete guide to GDPR compliance". [Online]. Available: https://gdpr.eu/. [Accessed: 25-Dec-2020].