Software Design Description Document for Intelligent Surveillance system for smart cities

Nour Ahmed, Nour El Hoda Hisham, Mariam Hesham, Samiha Hesham ,Sandra Fares
Supervised by: Eng. Lobna Mostafa, Dr. Islam Tharwat

March 31, 2021

Table 1: Document version history

Version	Date	Reason for Change
1.0	25-Mar-2021	SDD first version's description are defined.
1.1	26-Mar-2021	Added Sequence Diagram.
1.2	27-Mar-2021	Requirement Matrix updated.
2.0	29-Mar-2021	Diagrams updated.

GitHub: https://github.com/SandraFW/intelligent-Surveillance-for-smart-cities

Contents

Abstract

Surveillance systems are of vital importance for the development of smart cities. These systems can be considered vision organs of such cities. It is expected that a huge amount of data (Big Data) will be generated in smart cities. Therefore, to ensure the safety of its citizens, it is important to provide an efficient and real-time analysis of these data to get real-time responses, when catastrophic events occur. Accordingly, transmitting this massive data to the cloud, to be processed, is relatively slow. Therefore, the purpose of this project is to implement an edge computing- based surveillance system to offer real-time data processing. When surveillance videos capture an incident, the data get transferred to the edge for processing. Moreover, a rapid response is then provided to properly handle the occasion. Furthermore, despite tackling scalability obstacles, the system should handle privacy-sensitive data to overcome the privacy challenges in smart cities.

1 Introduction

1.1 Purpose

The purpose of the software design document (SDD) is to present the project " Intelligent Surveillance for smart cities" thoroughly. SSD illustrates the framework and system design architecture of the proposed system. Moreover, the document provides a fully detailed system overview and scope besides stating the main objective of the system. The SDD document is considered a significant guide for the project team members and for the customer to be fully aware of the project. Also, for the developers to improve and develop the system in the future.

1.2 Scope

The scope of the software design document (SDD) is to describe the project " Intelligent Surveillance for smart cities" in detail. The SDD provides the main design of the data and viewpoints to communicate to key design stakeholders. Also, it states the main objective of the system with a well-studied timeline. Besides, the algorithms and human Interface design that utilized in the project. Moreover, the states of the functional requirements of the system.

1.3 Overview

The software design document (SDD) presents the overall system architecture in addition to design patterns, data models, data structures, algorithms, and design viewpoints. Moreover to the human interface design to know how to utilize the provided services correctly. The software design document is structured as follows.

Section I introduces the document by defining its purpose, scope, and overview. In addition to the audience that the document is intended for. Section II presents the system thoroughly by illustrating the system overview and describing the functionality of the whole system. In addition to the system scope and brief description of system main features and boundaries. Moreover, it states the main objective of the system with the project plan. Section III illustrates the design viewpoints of the system. Section Iv presents the data design for the system. It explains the

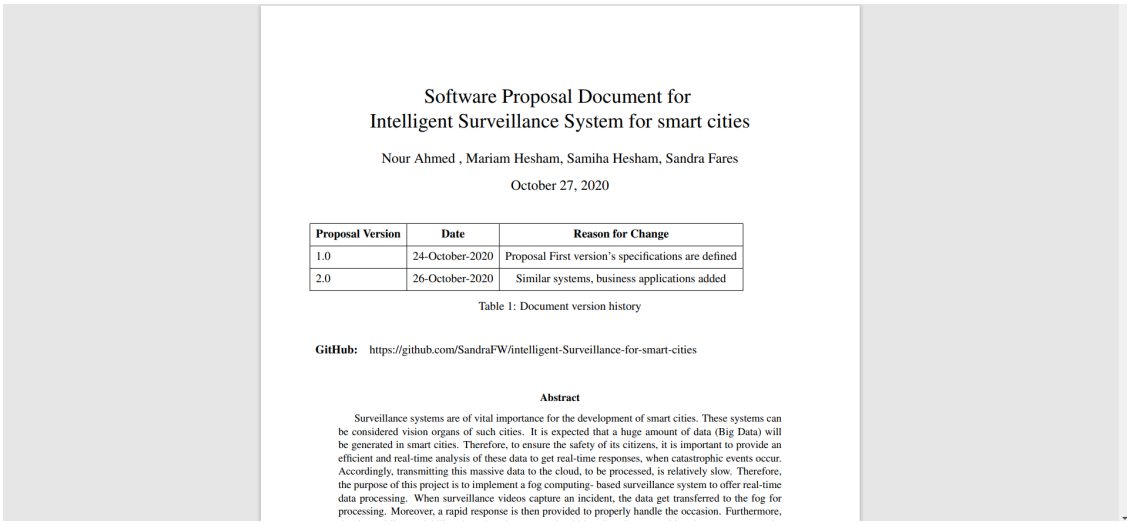
information related to the data in the system and how it is stored. In addition to the thorough description of the dataset and database utilized in the system. Section V presenting the human interface design for the system. Section VI describes the requirements matrix that clarifies the status of the functional requirements mentioned in the SRS. Section VII stating the appendices that could help in well understanding the document.

1.4 Intended audience

- Stakeholders: Dr. Islam Tharwat, Eng. Lobna Shaheen, Nour Ahmed Ghoneim, Samiha Hesham, Nour El-Hoda Hesham, Mariam Hesham, Sandra Fares.
- User: Police Officers, Admin
- Clients: Police, Security Government Companies, El SEWEDY technology company.

1.5 Reference Material

We have done a SRS which is mainly talking about how to use the edge-fog model and Proposal to handle abnormal movements in streets like kidnapping, fighting, and robbery. Moreover, we have done a proposal which is talking about big data and how to handle it by saving important data only and increase the security of this data from any massive or malicious action.



Software Proposal Document for
Intelligent Surveillance System for smart cities

Nour Ahmed , Mariam Hesham, Samiha Hesham, Sandra Fares

October 27, 2020

Proposal Version	Date	Reason for Change
1.0	24-October-2020	Proposal First version's specifications are defined
2.0	26-October-2020	Similar systems, business applications added

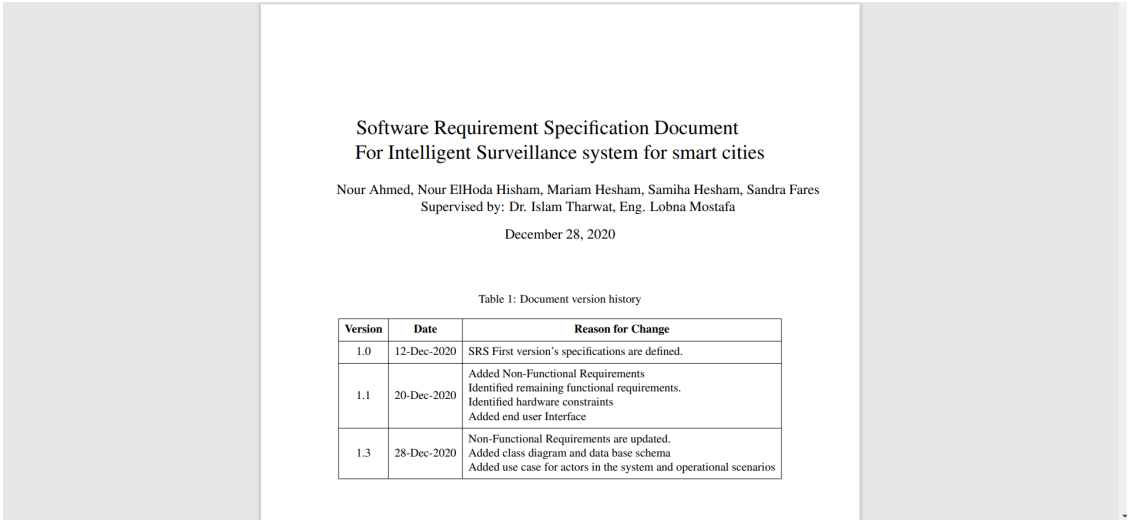
Table 1: Document version history

GitHub: https://github.com/SandraFW/intelligent-Surveillance-for-smart-cities

Abstract

Surveillance systems are of vital importance for the development of smart cities. These systems can be considered vision organs of such cities. It is expected that a huge amount of data (Big Data) will be generated in smart cities. Therefore, to ensure the safety of its citizens, it is important to provide an efficient and real-time analysis of these data to get real-time responses, when catastrophic events occur. Accordingly, transmitting this massive data to the cloud, to be processed, is relatively slow. Therefore, the purpose of this project is to implement a fog computing- based surveillance system to offer real-time data processing. When surveillance videos capture an incident, the data get transferred to the fog for processing. Moreover, a rapid response is then provided to properly handle the occasion. Furthermore,

Figure 1: Proposal



Software Requirement Specification Document
For Intelligent Surveillance system for smart cities

Nour Ahmed, Nour ElHoda Hisham, Mariam Hesham, Samiha Hesham, Sandra Fares
Supervised by: Dr. Islam Tharwat, Eng. Lobna Mostafa

December 28, 2020

Table 1: Document version history

Version	Date	Reason for Change
1.0	12-Dec-2020	SRS First version's specifications are defined.
1.1	20-Dec-2020	Added Non-Functional Requirements Identified remaining functional requirements. Identified hardware constraints Added end user Interface
1.3	28-Dec-2020	Non-Functional Requirements are updated. Added class diagram and data base schema Added use case for actors in the system and operational scenarios

Figure 2: SRS

1.6 Definitions and Acronyms

Term	Definition
EFISS	It stands for Edge Fog intelligent surveillance system.
Graphical User Interface (GUI)	Is a form of user interface that allows user to interact with electronic devices through graphical icons and audio indicator .
ISS	It stands for Inteligent Surveillance System
Deep Learning	Is a subfield of machine learning concerned with algorithms
Archive	Stores data that's no longer day to day use.
Backup	Designed for rapid recovery of operational data.

2 System Overview

2.1 System Scope

Our main objective is to build a fog computing based framework for surveillance data in order to detect abnormal human behaviors in the streets(fighting,kidnapping,robbery). The system is also

developed to work on enhancing the latency by reducing the amount of data. Moreover, we will work on developing the privacy protection scheme for data security.

1. Fog/Edge computing-based surveillance system to monitor the whole city 24/7.
2. Privacy assurance by blurring people's faces to secure their exposed identity in the video.
3. Scalability system to handle such massive data generated from surveillance cameras and sensors.
4. Efficient image processing techniques for certain failures in the edge cameras that impacts the usefulness of the video streams.
5. Dispatch crime details (i.e., video and location of the event, criminal's face) and giving alert to the nearest competent authority.

2.2 System objectives

1. The system will be able to detect the action occurred accurately.
2. The system should apply blurring method for normal citizen's bio-metrics in order to enhance privacy in surveillance system.
3. The system should send alert if any failure is detected in the cameras in order to help the maintenance team to solve the problem.
4. The system will send alert message when anomaly behaviour is detected using the fog nodes in the architecture system.
5. The system will filter the incident using video splitting technique in order to be saved and sent to the cloud to save storage.
6. The system should provide face recognition feature for the kidnapper and the victim to get their information.
7. The system ensures usability to achieve the demand efficiently with low latency and high bandwidth.
8. The Fog and edge models will reduce transmitted data which makes it easier for the system to reprocessing it, increase usability, and increase the potential of the cloud platform.

2.3 System Timeline

Table 2: ISS time plan

Id	Task	Start Date	Number of Days	Team Member
1	Collecting Dataset	2/1/2021	5	nour ahmed, sandra
2	Work on GUI	3/1/2021	15	samiha, mariam
3	camera failure	3/5/2021	10	nour elhoda
4	video summarization	3/1/2021	10	nour ahmed
5	crime detection	3/1/2021	10	sandra
6	face blurring technique	3/20/2021	20	mariam
7	face detection	3/20/2021	20	samiha
8	deep learning detection classifier	4/5/2021	15	nour ahmed,sandra
9	mobile application	4/5/2021	20	samiha,mariam
10	enhancing video quality	4/5/2021	10	nour elhoda

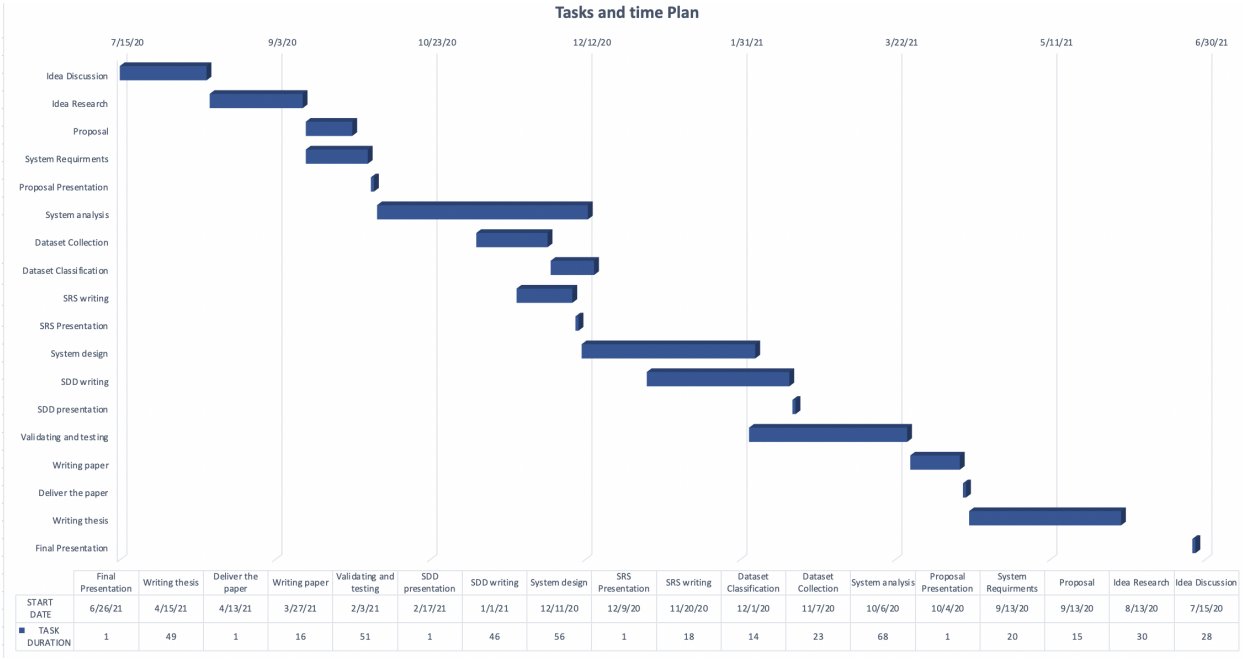


Figure 3: ISS GANTT Chart

3 Design viewpoints

3.1 Context viewpoint

The intelligent surveillance system monitors smart city streets to assists the police units to get notified of any incident that occurs to take quick action against it using the three computing layers Edge, Fog and cloud. Our proposed system aims to provide a high level of safety to the citizens. Therefore the system captures the pedestrians to collect data throughout the day. The collected data are processed to detect any failure in the surveillance cameras or whenever an abnormal activity occurs, it captures the incident, then identifies whether it is a crime or not; if it is, the system identifies the criminal and forwards the collected information to the police unit to take the proper action and helps them track the criminal.

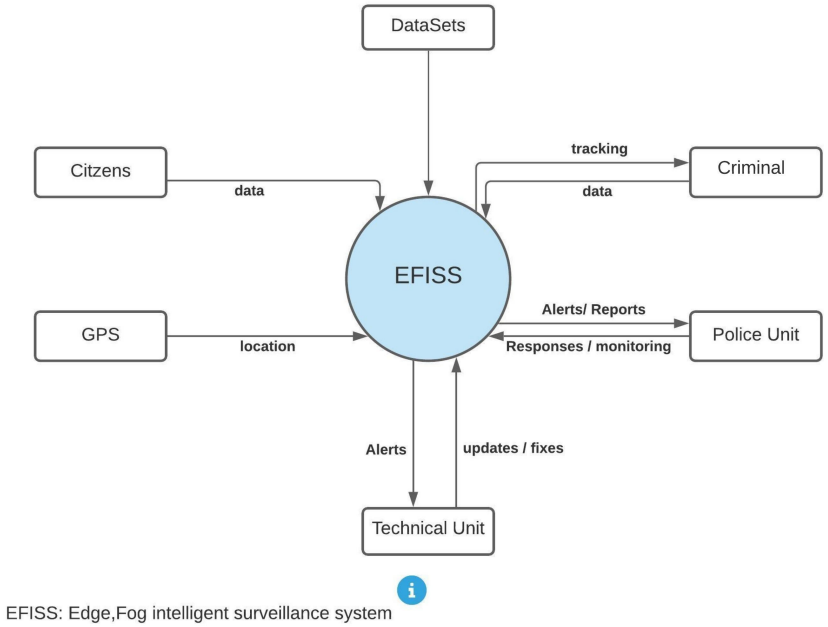


Figure 4: Context Diagram for the Edge Fog intelligent surveillance system

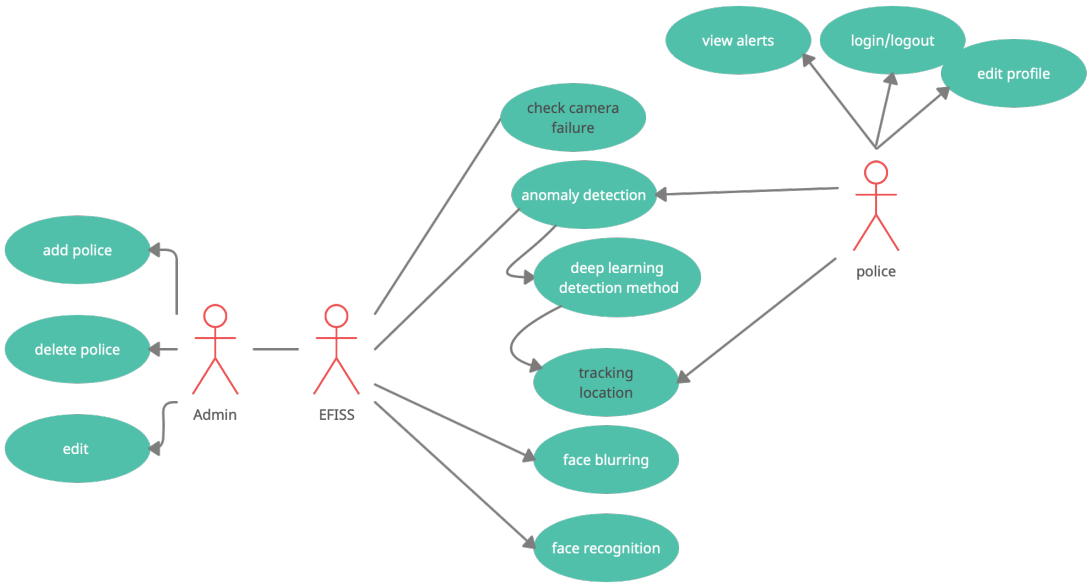


Figure 5: Use Case Diagram Example

3.2 Composition viewpoint

Our system is composed of three layers Edge, Fog, and cloud layers; Each of them has responsibilities, and they communicate together through a stable network to provide the user at the end the needed result.

- Edge Layer: The first layer in our system is the surveillance cameras fixed within the smart street lights. Those cameras are responsible for capturing the pedestrians throughout the day and using a crime detection algorithm whenever any abnormal actions are recognized the video stream is transferred to our second layer; and if everything is normal, the videos are being stored on the camera's local storage and then sent to the cloud when the network is free. Also, we consider any failure in cameras using a failure detection algorithm to avoid any blurry recording, green or black videos being recorded. Finally, there is a comparing algorithm that helps tracking wanted criminals whenever detected in any street the authorities are informed.

- Fog Layer: The second layer in our system, This layer is divided into two sub-layers; the first depends on connected vehicles near the incident and the second is smart traffic light which works as gateways.

  The vehicles layer: It is responsible for extracting more features from the video stream to be more sure about the incident and summarize it into an image using a special algorithm for further processing in the second fog sub-layer.

  The smart traffic light: It is responsible for the final processing of the incident using a deep learning algorithm; to ensure the incident, classify it and send it to the cloud layer the final layer. Also, an image processing deblurring algorithm is applied for face detection. then the criminal face image is sent back to the edge nodes.

- Cloud Layer: The final layer; is responsible for triggering the alarms and send notifications for our end users, and backup all the incident data sent. Also, it is used for archiving all the videos captured throughout the day.
- All the layers should communicate together using a stable wireless network provided in the smart city.

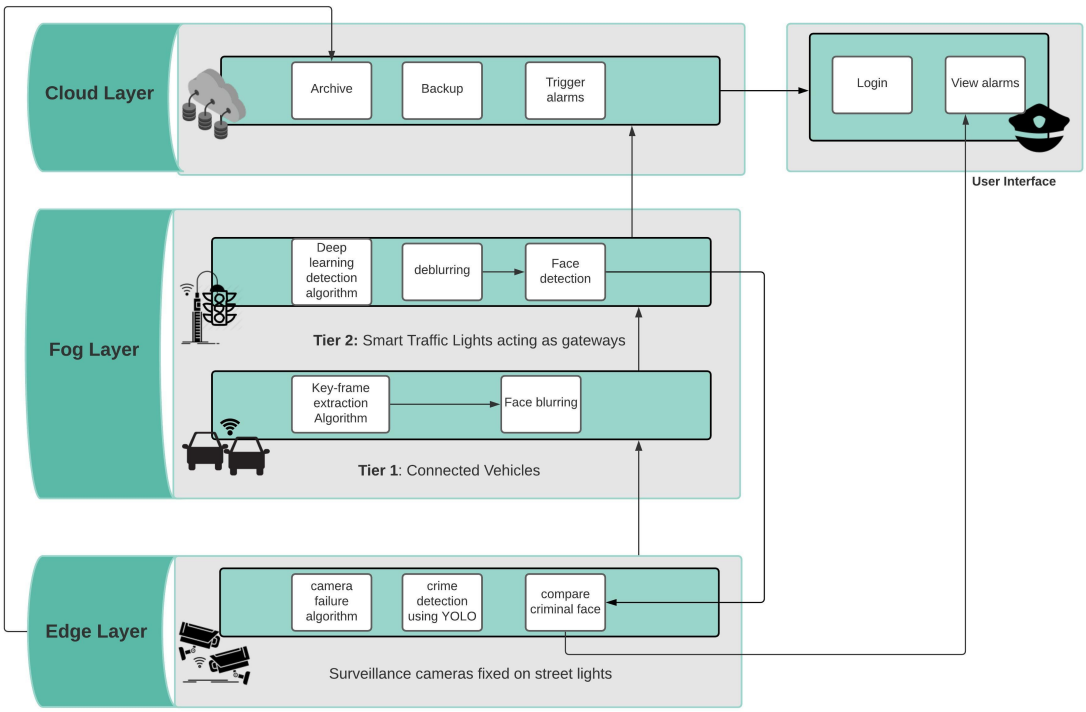


Figure 6: Architectural Design Diagram

Design concerns: Main design concerns are focused on robust and efficient communication between each computing layer. We put into consideration the computational power of devices used in each layer to be compatible with the functions deployed to them. performance of the functions, liable deep learning operation results, and avoiding false-negative detection also was of the main concerns to avoid false alarms.Finally, the system should follow ethics and protects the privacy of the citizens.

3.2.1 Design Rationale

Various architectures have been introduced in the field of real-time video surveillance.

- Cloud Computing: At first, systems were implemented using only a cloud computing layer; But they were expensive, relatively slow to perform real-time video surveillance, and there was consistently heavy traffic on the network.
- Edge or Fog layers and cloud Computing : Therefore, they started adding fog layers or edge layers individually, where the performance was enhanced; but it was introduced for

smaller areas as smart homes, also the devices used needed high computational power or less powerful functions were used.

So, We built our architecture that comprises all three layers to get maximum advantages from each layer; to easily monitor the whole smart city. Each layer performs specific functions either simple image processing or deep learning algorithms, therefore most of the false alarms were reduced; those functions were chosen to be deployed according to each layer's components' computational power and to efficiently serve the proper flow of the system. Then each layer communicates with others through a stable network to forward and process the results in case of detecting an incident only; thus the network traffic was reduced. And finally notifying our end users of those incidents efficiently.

3.3 Logical viewpoint

3.3.1 Class Diagram

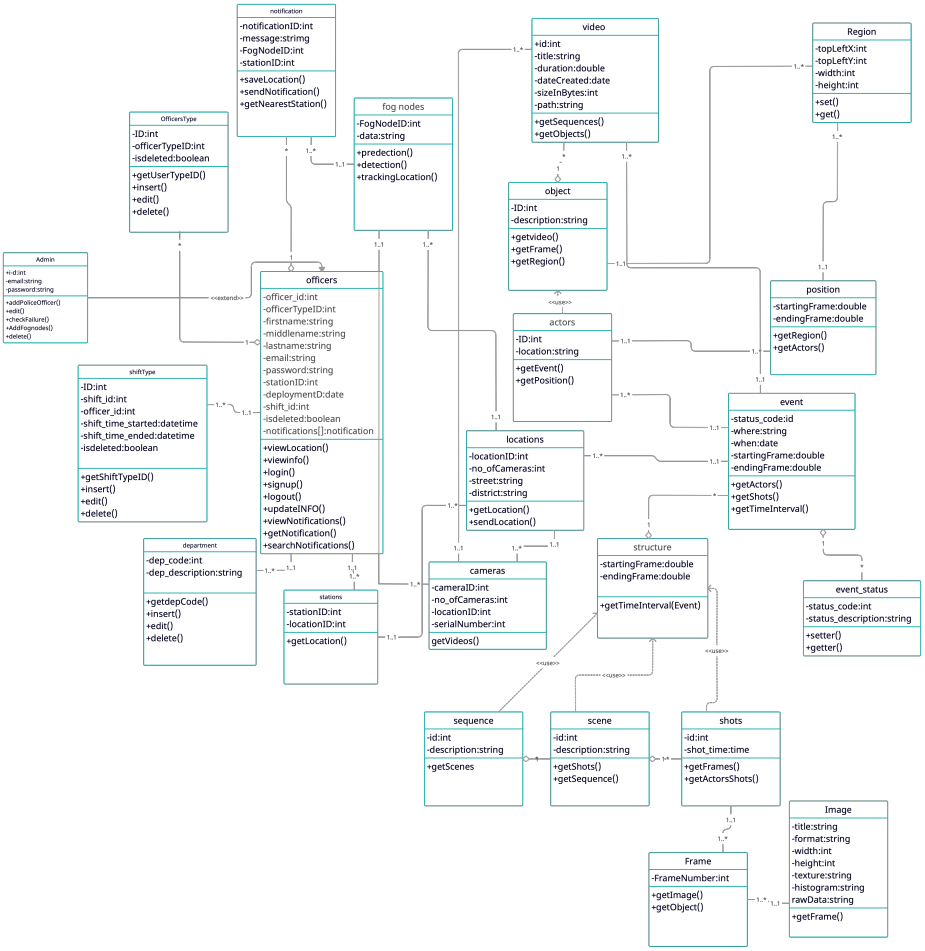


Figure 7: ClassDiagram

3.3.2 Detailed classes

Table 3: Class Name - officers

List of Super-classes	none
List of Sub-classes	none
Purpose	The class represents police officer'data
Collaborations	associated with shift type,stations,department and aggregated with officers type and notifications.
Attributes	officerid:int,officerTypeID:int,firstname:string,last name:string,email:string, password:string,stationID:int,deployment date:date,shiftID:int,isdeleted:boolean, notifications[]:notification.
Operations	login(email,password),logout(),signUP(firstname,lastname,email,password, officersTypeID),updateINFO(officersID,firstname,lastname,email,password), viewNotifications(notificationID),searchNotifications().
Constraints	officer is one of the core classes of the application

Table 4: Class Name - Officers type

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents officer's types.
Collaborations	aggregated with officers.
Attributes	ID:int,officerTypeID:int,isdeleted:boolean
Operations	getusertypeID(),insert(),edit(),delete()
Constraints	cannot work without officers class

Table 5: Class Name - Admin

List of Super-classes	officers
List of Sub-classes	none
Purpose	a class represent admin in the system
Collaborations	extends with officers.
Attributes	ID:int,email:string,password:string
Operations	createpoliceofficer(),edit(),delete(),checkFailure(),AddFognodes()
Constraints	Admin is the main controller in the application.

Table 6: Class Name - shift Type

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents shift types for the police officers.
Collaborations	associated with officers.
Attributes	ID:int,shiftID:int,officerID:int,shiftTimeStarted:datetime,shiftTimeEnded :date-time,isdeleted:boolean
Operations	getShiftTypeID(),insert(),edit(),delete()
Constraints	can't work without officers.

Table 7: Class Name - department

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents departments of the police officers.
Collaborations	associated with officers.
Attributes	depCode:int,depDescription:string
Operations	getDepCode(),insert(),edit(),delete()
Constraints	cannot work without officers.

Table 8: Class Name - stations

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents stations locations of each police officer.
Collaborations	associated with officers and locations.
Attributes	stationID:int,locationID:int
Operations	getLocation()
Constraints	cannot work without officers and locations.

Table 9: Class Name - notifications

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents the notifications that views to the police officer.
Collaborations	associated with fog nodes and aggregated with officers.
Attributes	ID:int,message:string,FogNodeID:int
Operations	saveLocation(),sendNotification(),getNearestStation() getLocation()
Constraints	cannot work without officers and fognodes.

Table 10: Class Name - fog nodes

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents the fognodes that sends the alerts that came from the edge node to the officers through class notifications
Collaborations	associated with notifications,locations and cameras.
Attributes	FogNodeID:int,data:string
Operations	prediction(),detection(),trackingLocation()
Constraints	cannot work without locations and cameras.

Table 11: Class Name - cameras

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents the edge nodes that carries the video data and ready to be sent to fog nodes.
Collaborations	associated with locations,fognodes and videos.
Attributes	cameraID:int,noOfcameras:int,locationID:int ,serialNumber:int
Operations	getVideos(),sendToFognodes(),caameraFailure(),CrimeDetection()
Constraints	cannot work without videos and locations.

Table 12: Class Name - locations

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents the locations of the surveillance camera that the kidnapping event will occur.
Collaborations	associated with cameras,stations,fogNodes and event.
Attributes	LocationID:int,noOfcameras:int,street:string,district:string
Operations	getLocation(),sendLocation()
Constraints	cannot work without event class

Table 13: Class Name - video

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents video data.
Collaborations	associated with cameras,event and aggregated with object. and event.
Attributes	id:int,title:string,duration:double,dateCreated:date,sizeInBytes:int,path:string
Operations	getSequences(),sendObjects()
Constraints	cannot work without object class.

Table 14: Class Name - object

List of Super-classes	none
List of Sub-classes	actors
Purpose	the class represents objects in the video frame.
Collaborations	associated with region and aggregated with video class.
Attributes	id:int,description:string
Operations	getVideo(),getFrame(),getRegion(),faceDetection(),faceBlurring
Constraints	cannot work without video class.

Table 15: Class Name - actors

List of Super-classes	object
List of Sub-classes	none
Purpose	the class represents actors which is mainly the kidnapper or the victim in the video frame.
Collaborations	associated with position and event.
Attributes	id:int,location:string
Operations	getEvent(),getPosition()
Constraints	cannot work without object class.

Table 16: Class Name - Region

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents the region of the objects that are in the frame video.
Collaborations	associated with position and object.
Attributes	topLeftX:int,topLeftY:int,width:int,height:int
Operations	getter(),setter()
Constraints	cannot work without object and position classes.

Table 17: Class Name - position

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents the exact video frame that the event occur at be saved and detection will be processed on it.
Collaborations	associated with region and actors.
Attributes	startingFrame:double,endingFrame:double
Operations	getRegion(),getActors()
Constraints	cannot work without actors class.

Table 18: Class Name - event

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents the event of kidnapping.
Collaborations	associated with video,actors,locations and aggregated with event status and structure
Attributes	statusCode:id,where:string,when:date,startingFrame:double,endingFrame:double.
Operations	getshots(),getActors(),getTimeInterval()
Constraints	cannot work without actors and locations classes.

Table 19: Class Name - event status

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents the status of the event occurred.
Collaborations	aggregated with event class.
Attributes	statusCode:id,statusDescription:string
Operations	getter(),setter()
Constraints	cannot work without event class.

Table 20: Class Name - structure

List of Super-classes	none
List of Sub-classes	scene,sequence,shot
Purpose	the class represents the structure of the event occurred that is divided into shots,scenes and sequence.
Collaborations	aggregated with event class.
Attributes	startingFrame:double,endingFrame:double
Operations	getTimeInterval(Event)
Constraints	cannot work without event class.

Table 21: Class Name - sequence

List of Super-classes	structure
List of Sub-classes	none
Purpose	the class represents the sequence of the structure the video surveillance captured when an event occured.
Collaborations	aggregated with scene class.
Attributes	id:int,description:string
Operations	getScenes()
Constraints	cannot work without structure class.

Table 22: Class Name - scene

List of Super-classes	structure
List of Sub-classes	none
Purpose	the class represents the scenes of the kidnapping event that occurred.
Collaborations	aggregated with shots class.
Attributes	id:int,description:string
Operations	getSequence(),getShots()
Constraints	cannot work without structure class.

Table 23: Class Name - shots

List of Super-classes	structure
List of Sub-classes	none
Purpose	the class represents the shots of the actors in the event that occurred.
Collaborations	associated with frame class.
Attributes	id:int,shotTime:time
Operations	getFrames(),getActorShots()
Constraints	cannot work without structure class.

Table 24: Class Name - Frame

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents the frame of the actors that occurred in an event.
Collaborations	associated with shot and image classes.
Attributes	FrameNumber:int
Operations	getImage(),getObject()
Constraints	cannot work without shot class.

Table 25: Class Name - Image

List of Super-classes	none
List of Sub-classes	none
Purpose	the class represents the image data of the frame captured.
Collaborations	associated with frame class.
Attributes	title:string,format:string,width:int,height:int,texture:string, histogram:string,rawData:string
Operations	getFrame(), deepLearningDetection(),videoSummarization()
Constraints	cannot work without Frame class.

3.4 Patterns use viewpoint

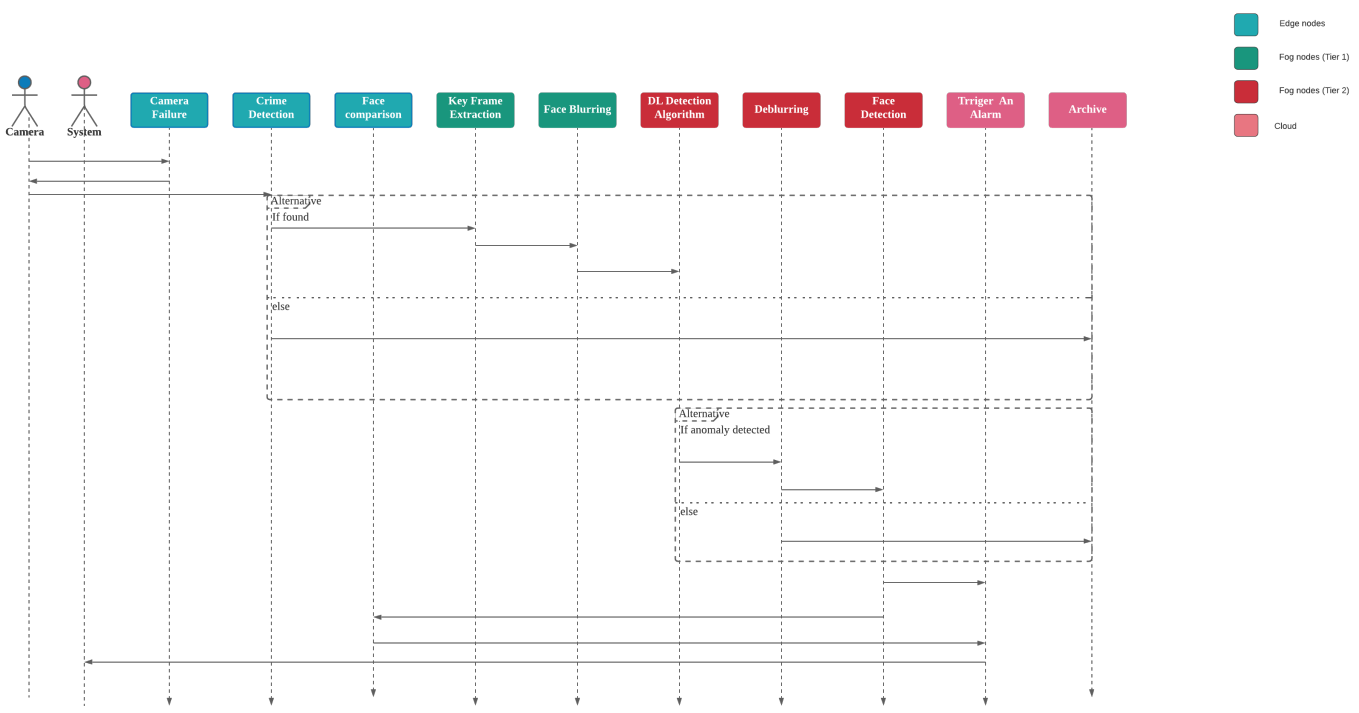


Figure 8: Sequence Diagram

3.4.1 Design Rationale

Observer Flutter Design Pattern:The observer pattern is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods which is used in our mobile application.

3.5 Algorithm viewpoint

The system consists of edge-fog layers, where each layer is responsible for a sequence of tasks that it completes and passes on to the next layer.

- Sensors placed on street lights are introduced in this system as our edge nodes. The edge nodes are responsible for checking for camera failures. They make sure that the camera capturing is not interrupted quality-wise by any external sources. Moreover, edge nodes apply a face comparison algorithm to compare faces detected with the faces detected in previous captures. Also, a crime detection algorithm using YOLO is applied to acknowledge if a crime occurred. When a crime is detected, the nodes determine the current location of the incident and send it, with the video streams, to the next layer.

- Vehicles are proposed as the system's nodes for the second layer, the first tier fog layer. The fog nodes in this layer are responsible for applying a video summarization approach which is the keyframe extraction technique. The goal of this approach is to decrease the computational power by eliminating repeated frames and choosing frames that have relevant information as keyframes. Moreover, a face blurring algorithm is applied to protect the privacy of the citizens.

- Summarized video frames are then passed to the second-tier fog layer, where the nodes are the traffic lights. Deep learning techniques are applied to determine if the contact between people was unproblematic or if an incident occurred. If a crime occurred, the algorithm determines the crime's type. Moreover, deblurring and face detection get applied to expose the criminal's face.

- The cloud layer, which is the last layer, collects all the data passed to it by the layers and triggers an alarm to police units. Moreover, it backs up the data received by the layers and contains an archive to store video streams from the edge layer.

3.6 Interaction viewpoint

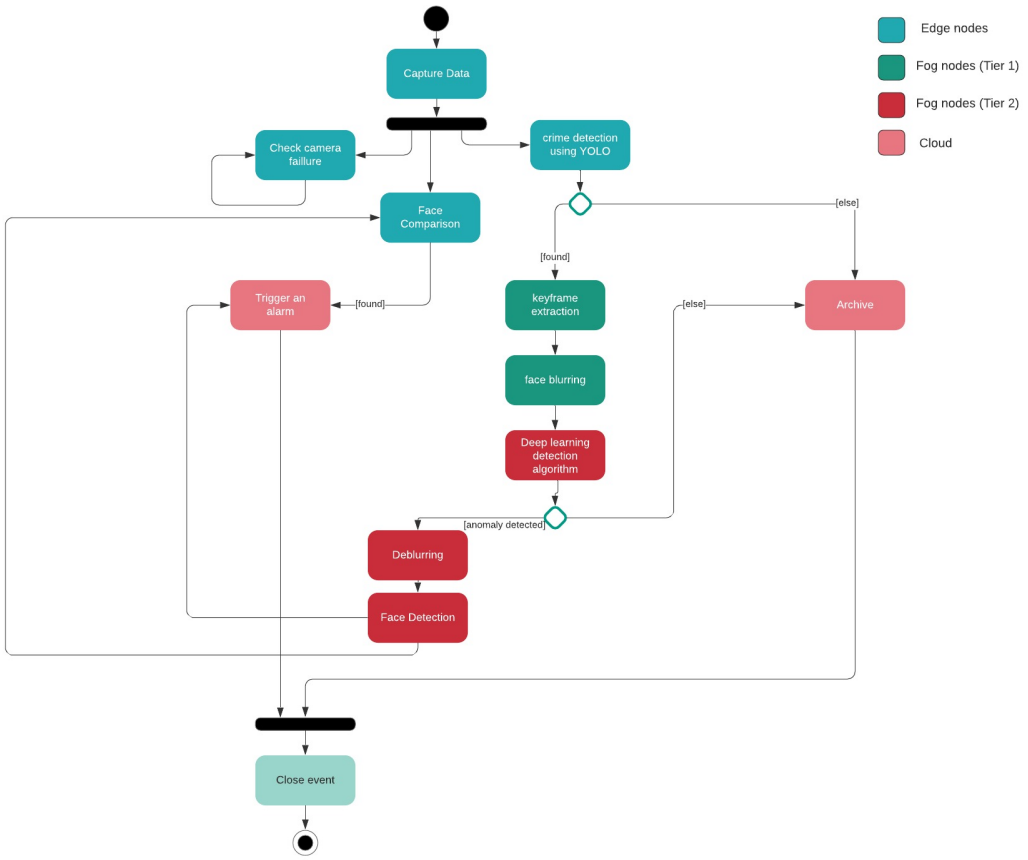


Figure 9: The workflow diagram of the framework

3.7 Interface viewpoint

3.7.1 Main Page

The admin can create accounts for multiple users. Each user logs into the system with his email and password. Moreover, each user can log out of the system.

- Login allows the user to enter the system using his email and password
- Registeration allows the admin to create account for the users.
- Logout allows the user to exit the system.

3.7.2 Location Page

The location page consists of a map which views the incidents location. Users can track incidents to take appropriate action.

3.7.3 Alert Page

The alert page consists of alerts triggered by the cloud layer. It has all the information collected needed by police units to track and capture criminals.

4 Data Design

4.1 Data Description

Our system handles and processes data in various ways as data needed are retrieved from either the application implemented for storing users' data or from the edge layer of our system's architecture. Moreover, Data received from our dataset to train our models is originally in CSV file format. Data get captured in the application from the forms available for the users to fill in. On the other hand, sensors (cameras) placed on the edge layer capture data in video format to get processed further. Information extracted from users will be stored in Cloud Firestore, a NoSQL database that will provide our application with flexibility and scalability.

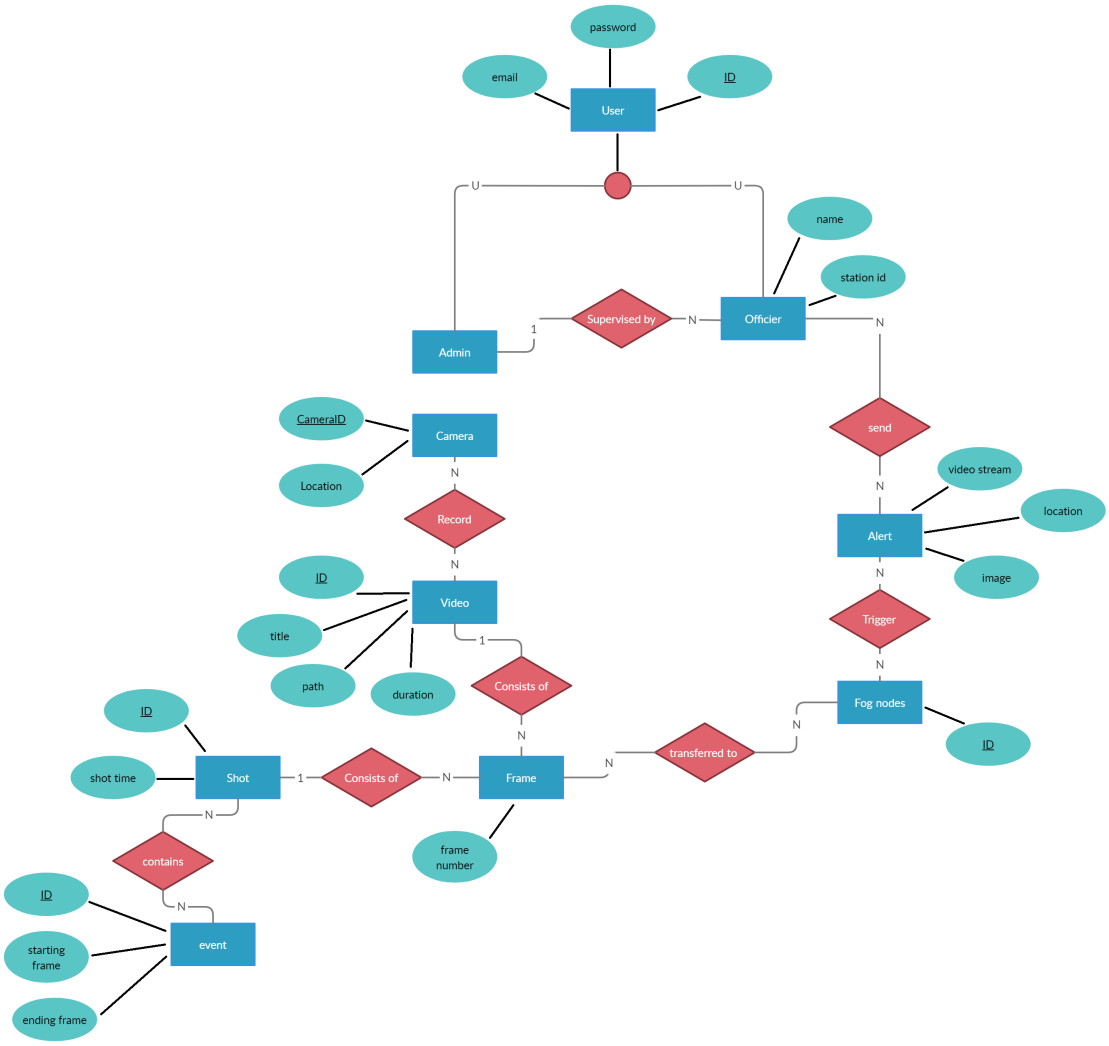


Figure 10: ER Diagram

4.2 Dataset Description

We have acquired a data set of surveillance videos published by the **'UCF Center for research in computer vision'** called **UCF-Crime**; It is a large-scale dataset of 128 hours of videos; it has 1900 video of many classes as Robbery, Stealing, Shooting ... etc. And it is widely used in any research or project that targets public safety and surveillance. Therefore we took some of those videos from the classes that meet our requirements. 4.2 illustrates the dataset.

Table 26: Dataset Description

Dataset Name	UCF-Crime Dataset
Link	https://www.crcv.ucf.edu/projects/real-world/
Size	95.9 GB
Number of classes	The dataset has 13 classes for anomaly events and normal class; we used 3 of them
Average Video frames	1000-1500 frames per video
Number of videos used	we used for fighting 10 videos,Robbery 10 videos,normal events 10 videos
comments	This data set has many videos in many places; we chose the suitable videos in streets. Also it didn't include kidnapping events; so we gathered 6 surveillance videos from Youtube with same video length and specifications as UCF-crime videos

4.3 Database design description

The admin is responsible for creating the officers accounts and later the officers has the ability to login and to see their alerts panel in case any incident is detected. The camera of each location is linked with the nearest police station.

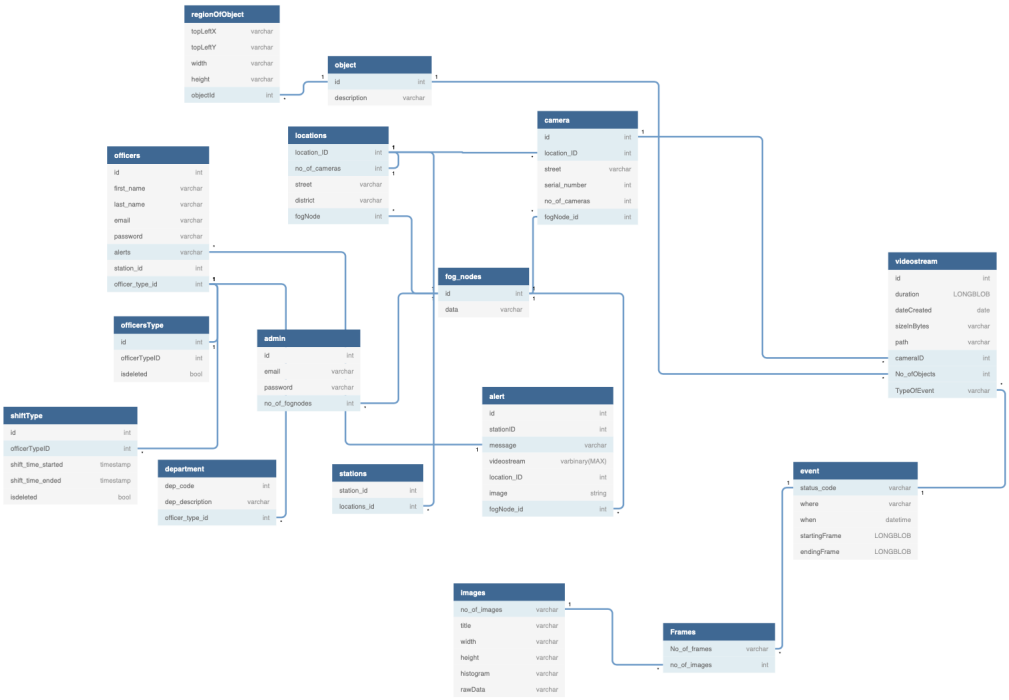


Figure 11: Database

5 Human Interface Design

5.1 User Interface

The proposed system's application focuses on two main ends. User end and admin end. The application allows the admin to login and creates accounts for the police officer or security guards. Also, the admin can update and delete the user's account. The user will log in to the application with an email and password set by the admin. Also, the user can update his information and change his password. Moreover, the user will view the alerts sent by the system with images, location, and time.

Admin: Admin will log in through the application login page. Admin can add accounts by filling the fields in add account page with proper information.

User: The user will log in through the application login page. He will be able to edit the profile by updating the user's info through the profile page. Also, the user will view the alerts sent by the system with their details on the alert page.

5.2 Screen Images

Figure 12: Home Page

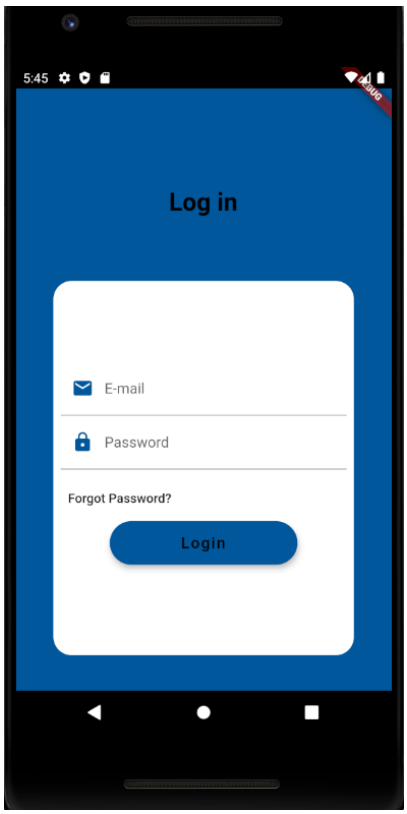


Figure 13: Login Page

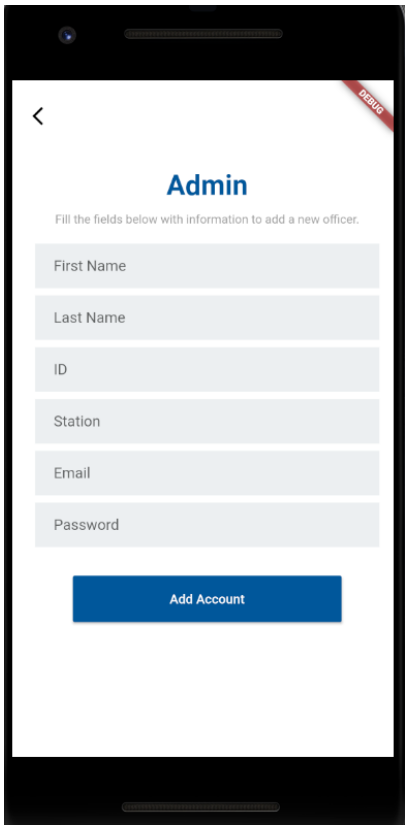


Figure 14: Add new account

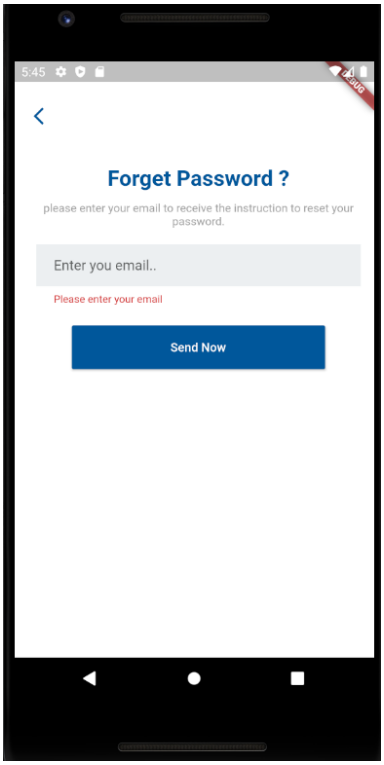


Figure 15: Forget Password

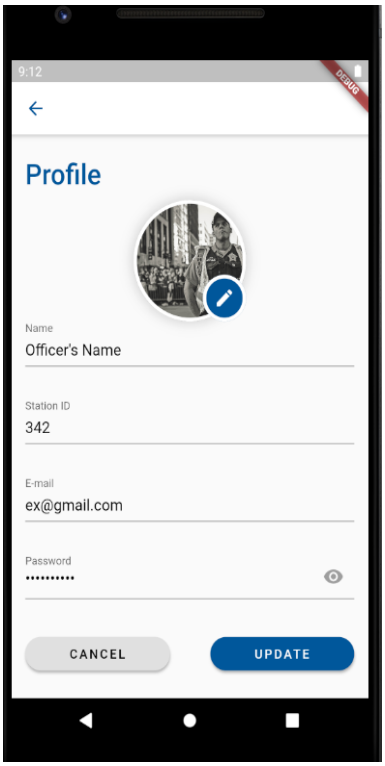


Figure 16: Profile

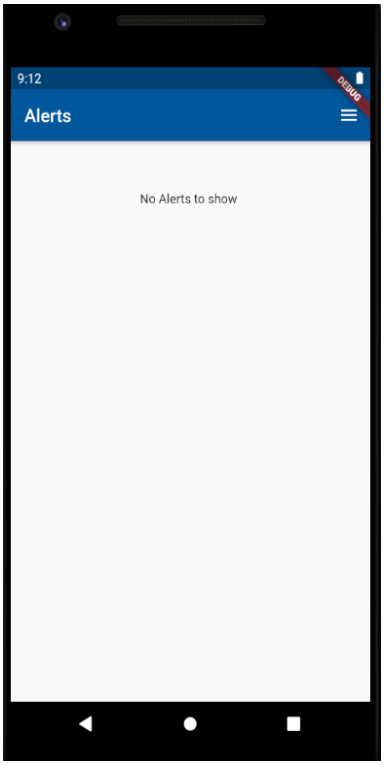


Figure 17: Alerts Page

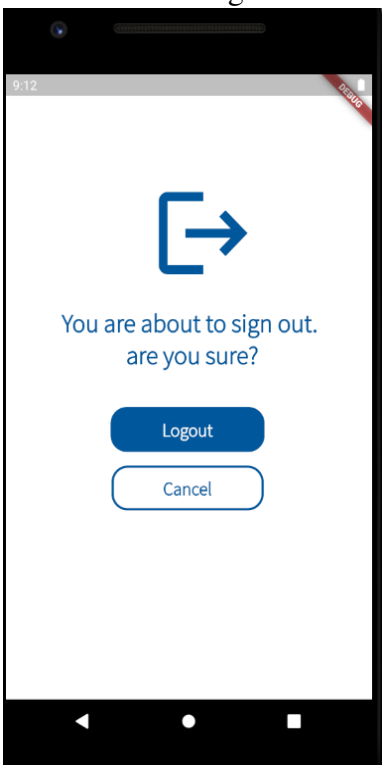


Figure 18: Logout Page

5.3 Screen Objects and Actions

Our proposed system has two main stakeholders, the admin will login by his email and password then he is responsible to create accounts for police officers or security guards also has the power to edit, delete user's accounts. Moreover, the second one is user, he will login with his email and password in the login page that was created by the admin, as he has the ability to edit or delete his information and can view alerts, location, and images sent by the system. Also, the user can make a new password if he forgets his password on forget password page.

6 Requirements Matrix

Table 27: Requirements Matrix

Req. ID	Req Desc	Class	Test Cases ID	Status
FR01	camera failure (edge layer)	camera	TC03, TC04, TC05	Developed
FR03	crime detection using YOLO (edge layer)	camera	TC01, TC02	Developed
FR04	video summarization (fog tier 1)	image	TC06	Developed
FR05	face blurring (fog tier 1)	object	TC07	Developed
FR06	deep learning detection (fog tier 2)	image	TC08, TC09	In Progress
FR07	face detection (fog tier 2)	object	TC10	Developed
FR09	login	officers	TC11,TC12	Developed
FR10	create account	Admin	TC13,TC14,TC15	Developed

7 APPENDICES

Fog computing: A mid-layer between edge devices and the cloud, at which data processing occurs.

Edge computing: An architecture at which data gets processed on devices rather than the cloud to provide low latency.

7.1 Github

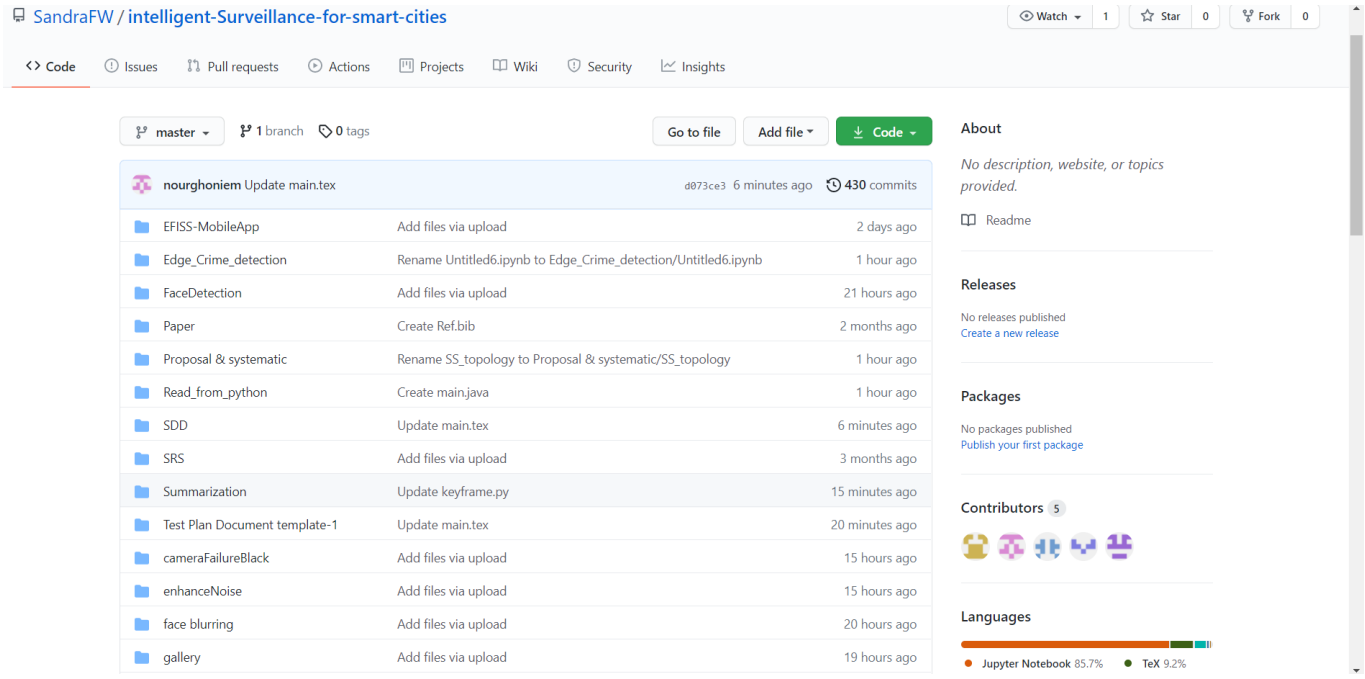


Figure 19: GitHub repository

7.2 Other appendices as appropriate

Optional section.