**COMSATS University Islamabad,   
Park Road, Chak Shahzad, Islamabad Pakistan**

SOFTWARE DESIGN DESCRIPTION   
(SDD DOCUMENT)

for

**DARTS**

Detection And Recognition with Tracking of Suspects  
Version 1.0

***By***

**Arbaz Ajaz CIIT/FA16-BCS-001/ISB**

**Kanwal Shariq CIIT/FA16-BCS-005/ISB**

***Supervisor*Dr. Tahir Mustafa Madni**

**Co-supervisor**

**Dr. Uzair Iqbal**

*Bachelor of Science in Computer Science (2016-2020)*

**Table of Contents**

Revision History iii

1. Introduction 1

Module 1: Cameras Manager 1

Module 2: Tier 1 Servers handling system 1

Module 8: User Management 1

Module 9: QR forces communication system 1

2. Design methodology and software process model 2

3. System overview 2

3.1 Architectural design 3

3.2 Process flow/Representation 4

4. Design models 13

4.1 Class diagram 13

4.2 Sequence diagrams 14

5. Data design 20

5.1 Data dictionary 22

6. Algorithm & Implementation 24

7. Software requirements traceability matrix 24

8. Human interface design 26

8.1 Screen images 26

8.2 Screen objects and actions 26

9. References 26

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

**Application Evaluation History**

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |

Supervised by

Dr. Tahir Mustafa Madni

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction

The following document contains the design specifications of the sub-systems of our application.

The application is introducing a fully automated system that will detect and recognize a suspect even if it is occluded and the last saved picture of him is of his childhood. It will also detect the objects it is carrying, track movement of the person of interest in real time and predict its next location. Based on the severity of the threat, it will alert the quick response forces through a mobile application that will also provide them with a communication interface. The designated authoritative figure will be able to monitor the movement through the web platform. The application is designed to assist the LEAs in quickly responding to the imminent threats that an individual pose and to also trace the initiation of his movements and apprehend the culprits before they get a chance to act.

We have implemented the interface and database modules that will make our work much easier as we will have the frontend and database already made and now, only have to work on image processing algorithms. Out of 10 modules, we have been able to implement the following 4 so far:

## Module 1: Cameras Manager

This module will be in charge of keeping track of all the cameras connected to the system by their geo locations. It will allow the admin to view the video feed of each camera and also to send it to the servers for further processing. The system will also allow the admin to delete or add a camera to the system which will be useful for the scalability of the system.

## Module 2: Tier 1 Servers handling system

To be safe from overload and bottleneck, this module will distribute the incoming camera feeds to the servers attached to the system through camera connections. It will check which server is free and will keep the video frames to be processed without interruption. If a suspect is recognized, it will send the required data to the main server. The system will also handle the addition and removal of servers.

## Module 8: User Management

The system has three registered actors. One is the designated authority figure who can use the system to monitor the teams, the other is the admin that manages the system through its admin privileges, and the last is the quick response team which will be registered by scanning the QR code on the system. Each team will have the application to receive alerts and communicate with other teams.

## Module 9: QR forces communication system

To provide a single and safe interface, our project will include a communication system for the quick response teams so they could be safe and can call for backup in need of emergency. This will be a mobile application that the QR forces will use that would include all their information and a map to show locations of all other QR teams and suspects.

# Design methodology and software process model

We will be using Object Oriented approach for data encapsulation, modularity and abstraction. The OOP approach will enable us to properly communicate between different modules in our program. Furthermore this method will ensure the proper and efficient use of programming with better understanding of the flow of program.

We will be using iterative and incremental build model where we will progress and improve incrementally; build more out of the prototype again and again. As the requirements may change, or there may be other changes at various stages of the project, the incremental model will provide the flexibility of making those changes

# System overview

This project presents a web and mobile application for the use of Law Enforcement Agencies, and other security related organizations, through which they will be able to monitor the movements of a suspect. Suspicious behavior will be monitored and suspect will be detected and recognized through facial recognition algorithms that will be running on the available surveillance camera’s feeds in real time. As soon as a suspect is recognized (through available pictures in the database however old) or a suspicious behavior is seen, the tier 1 servers will send the relevant data to the main server where it will begin mapping the suspect’s movements and predicting its next location based on machine learning algorithms. Based on the severity of the threat, the system will notify the Quick response units that are registered through QR code, through a mobile application and provide them with an interface to communicate with other units, formulate a plan and be provided with a backup. On the other hand, the system will provide a monitoring system for the authorized person so they can strategically capture the suspect.

Our system’s main features include:

* Facial recognition algorithms
* Machine learning algorithms
* Face recreation algorithms
* Abandoned object detection
* Path mapping of suspect
* Severity of threat calculation
* Mobile and Web application
* Preprocessing and enhancement of images
* Communication system
* Database management

.

## Architectural design

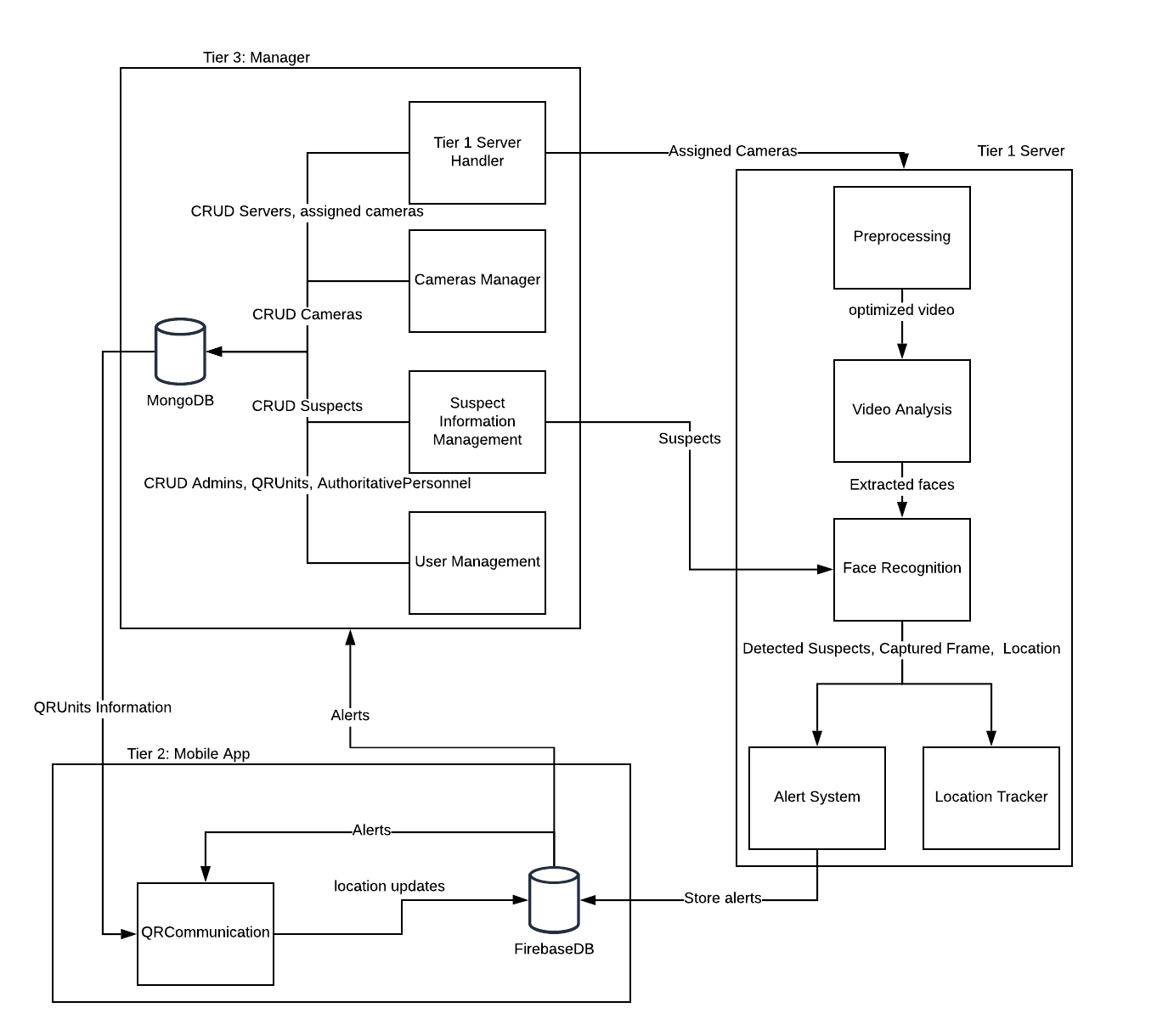
Our system has a tier architecture to improve horizontal scalability, performance and availability. Our system is divided into manager tier that manages servers, cameras and users, mobile application that responds to alerts and provides a communication interface and tier 1 servers that are responsible for detection, recognition of suspects.

Figure : Architecture diagram

## Process flow/Representation

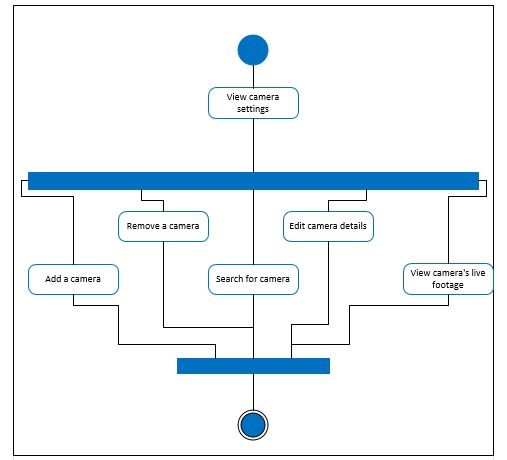


Figure : Activity diagram for Camera Manager

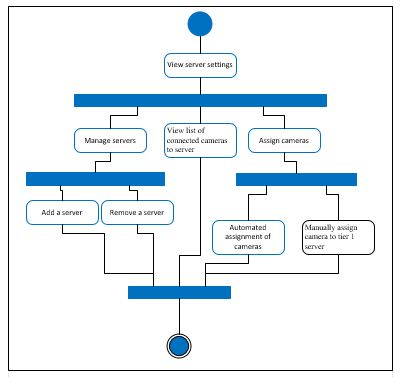


Figure : Activity diagram for Tier 1 server manager

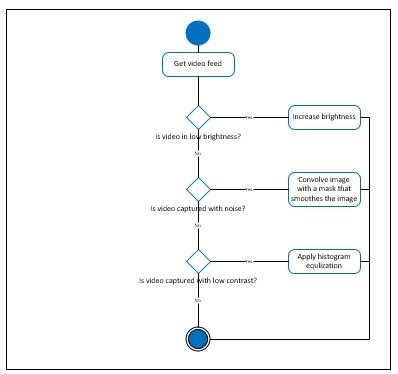


Figure : Activity diagram for Preprocessing video feed

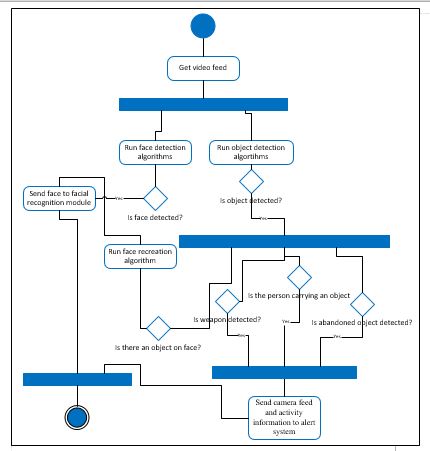


Figure : Activity diagram for Video analysis

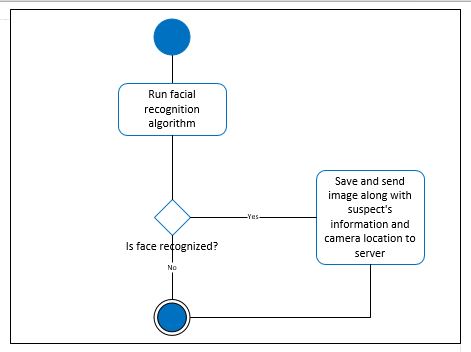
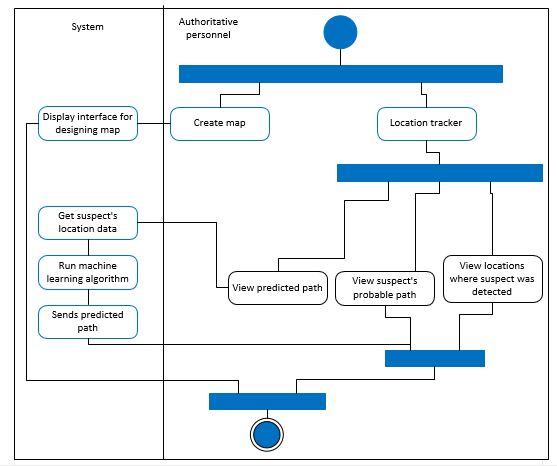


Figure 7: Activity diagram for Location tracker

Figure 6: Activity diagram for Face recognition

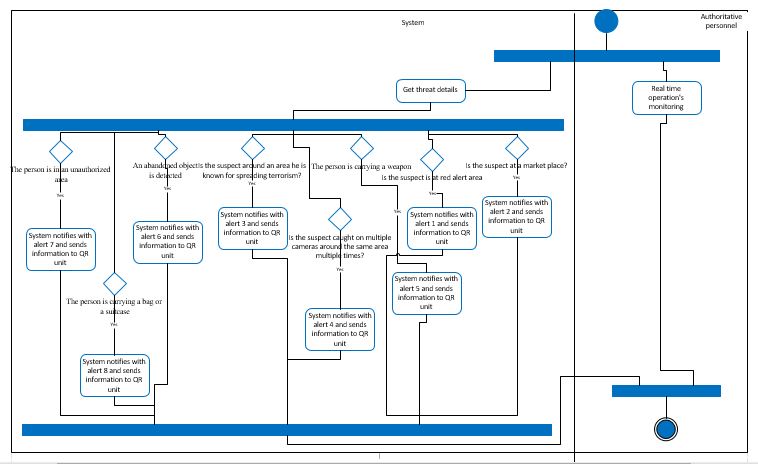


Figure 8: Activity diagram for Alert system

# 

Figure 9: Activity diagram for User management

# 

Figure 10: Activity diagram for QR communication system

# 

Figure 11: Activity diagram for Suspect information management

# Design models

## Class diagram

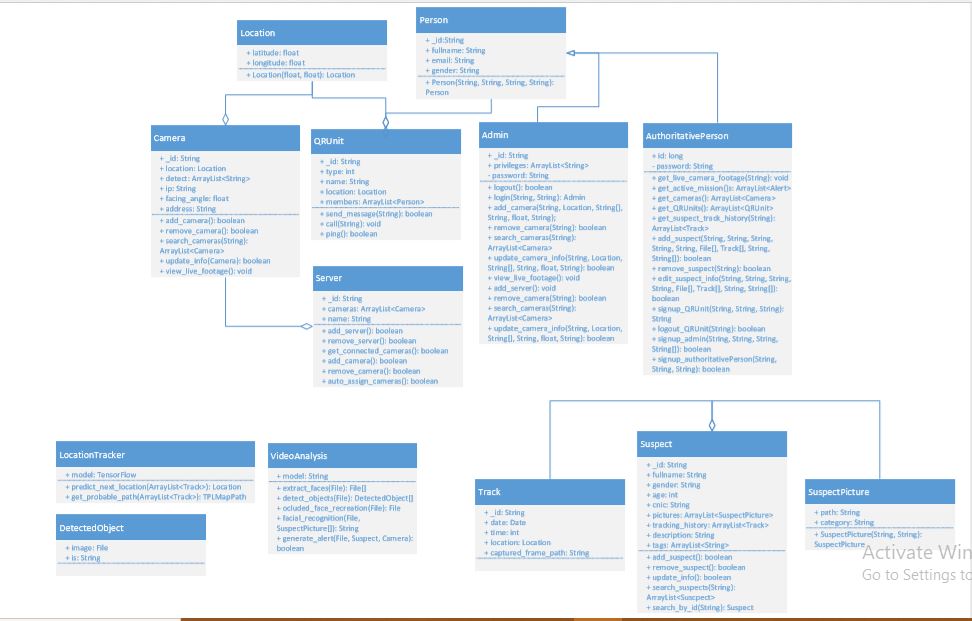


Figure 12: Class diagram

## Sequence diagrams

Figure 13: Sequence diagram for adding a server

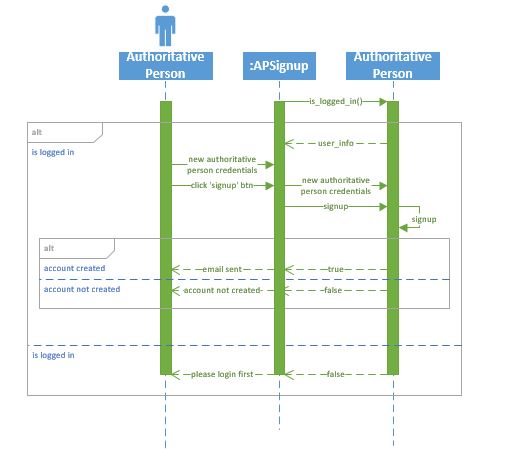


Figure 14: Sequence diagram for Authoritative person sign up

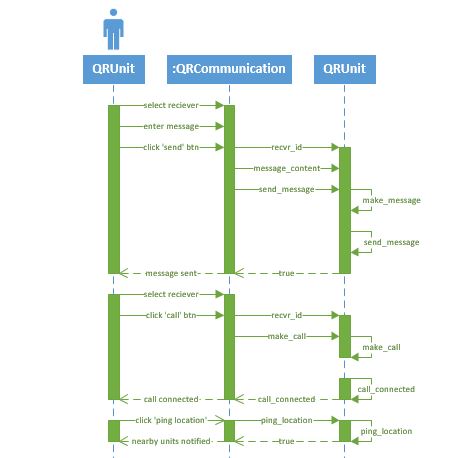
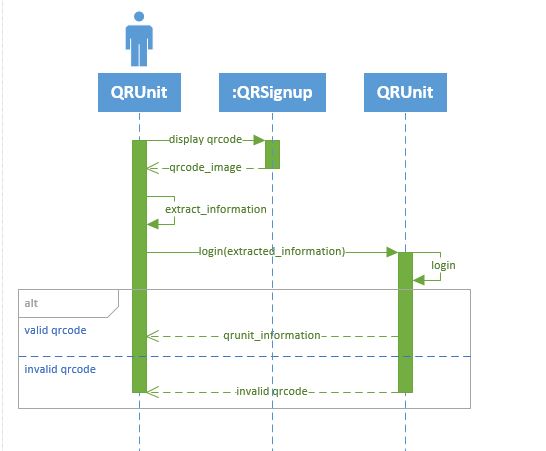


Figure 15: Sequence diagram for QR communication system



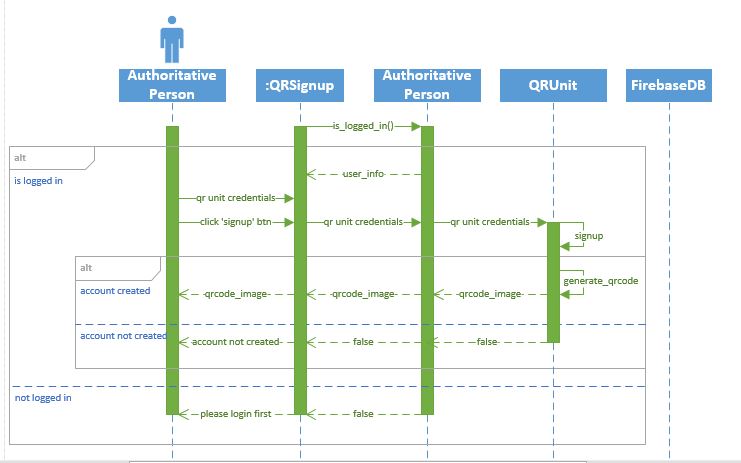


Figure 17: Sequence diagram for QR login

Figure 16: Sequence diagram for QR sign up

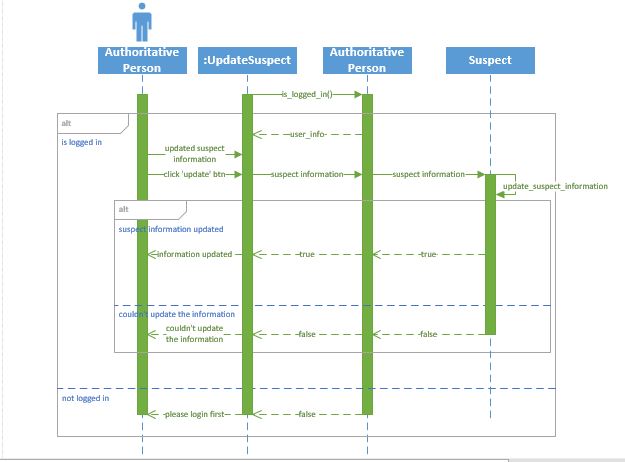


Figure 18: Sequence diagram for updating suspect information

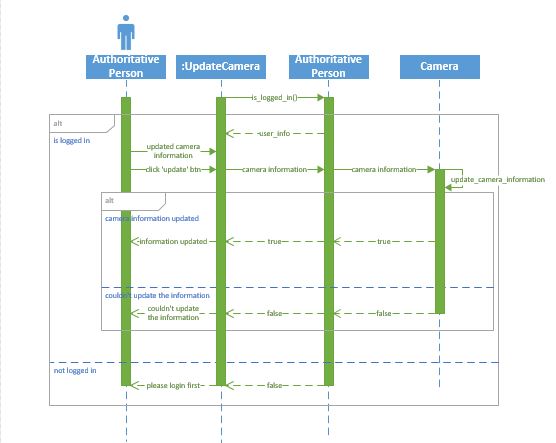


Figure 19: Sequence diagram for updating camera information

# Data design

As we are using Firebase and MongoDB to handle data, we have stored data into a JSON Schema.

**JSON Schema:**

var Admin = new Schema({

password: {

type: String,

required: true

},

privileges: {

type: [String],

required: true

},

person: {

type: Schema.Types.ObjectId,

required: true

}

})

var AuthoritativePerson = new Schema({

password: {

type: String,

required: true

},

person: {

type: Schema.Types.ObjectId,

required: true

}

})

ar Camera = new Schema({

longitude: {

type: Number,

required: true

},

latitude: {

type: Number,

required: true

},

ip: {

type: String,

required: true

},

facing\_angle: {

type: Number,

required: true

},

address: {

type: String,

required: true

},

Detect: {

type: [String],

required: true

}

})

var Person = new Schema({

fullname: {

type: String,

required: true

},

gender: {

type: String,

required: true

},

email: {

type: String,

required: true

}

})

var QRUnit = new Schema({

type\_of: {

type: String,

required: true

},

title: {

type: String,

required: true

},

members: {

type: [Schema.Types.ObjectId],

required: true

}

})

var Server = new Schema({

cameras: {

type: [Schema.Types.ObjectId],

required: true

},

name: {

type: String,

required: true

}

})

var Suspect = new Schema({

fullname: {

type: String,

required: true

},

gender: {

type: String,

required: true

},

age: {

type: Number,

required: true

},

pictures: {

type: [Schema.Types.ObjectId],

required: true

},

tracking\_history: {

type: [Schema.Types.ObjectId],

required: true

},

description: {

type: String,

required: true

},

tags: {

type: [String],

required: true

}

})

var SuspectPicture = new Schema({

path: {

type: String,

required: true

},

category: {

type: String,

required: true

}

})

## Data dictionary

**Person:**

+ Person(String, String, String, String): Person

**Location:**

+ Location(float, float): Location

**Camera:**

+ add\_camera(): boolean

+ remove\_camera(): boolean

+ search\_cameras(String): ArrayList<Camera>

+ update\_info(Camera): boolean

+ view\_live\_footage(): void

**QR unit:**

+ send\_message(String): boolean

+ call(String): void

+ ping(): Boolean

**Admin:**

+ logout(): boolean

+ login(String, String): Admin

+ add\_camera(String, Location, String[], String, float, String);

+ remove\_camera(String): boolean

+ search\_cameras(String): ArrayList<Camera>

+ update\_camera\_info(String, Location, String[], String, float, String): boolean

+ view\_live\_footage(): void

+ add\_server(): void

+ remove\_camera(String): boolean

+ search\_cameras(String): ArrayList<Camera>

+ update\_camera\_info(String, Location, String[], String, float, String): boolean

**Authoritative Person:**

+ get\_live\_camera\_footage(String): void

+ get\_active\_mission()s: ArrayList<Alert>

+ get\_cameras(): ArrayList<Camera>

+ get\_QRUnits(): ArrayList<QRUnit>

+ get\_suspect\_track\_history(String): ArrayList<Track>

+ add\_suspect(String, String, String, String, String, File[], Track[], String, String[]): boolean

+ remove\_suspect(String): boolean

+ edit\_suspect\_info(String, String, String, String, File[], Track[], String, String[]): boolean

+ signup\_QRUnit(String, String, String): String

+ logout\_QRUnit(String): boolean

+ signup\_admin(String, String, String, String[]): boolean

+ signup\_authoritativePerson(String, String, String): Boolean

**Server:**

+ add\_server(): boolean

+ remove\_server(): boolean

+ get\_connected\_cameras(): boolean

+ add\_camera(): boolean

+ remove\_camera(): boolean

+ auto\_assign\_cameras(): Boolean

**LocationTracker:**

+ predict\_next\_location(ArrayList<Track>): Location

+ get\_probable\_path(ArrayList<Track>): TPLMapPath

**VideoAnalysis:**

+ extract\_faces(File): File[]

+ detect\_objects(File): DetectedObject[]

+ ocluded\_face\_recreation(File): File

+ facial\_recognition(File, SuspectPicture[]): String

+ generate\_alert(File, Suspect, Camera): Boolean

**Suspect:**

+ add\_suspect(): boolean

+ remove\_suspect(): boolean

+ update\_info(): boolean

+ search\_suspects(String): ArrayList<Suscpect>

+ search\_by\_id(String): Suspect

**SuspectPicture:**

+ SuspectPicture(String, String): SuspectPicture

# Algorithm & Implementation

We have implemented **TPL maps API** for navigation, tracking and location pinging in both web and mobile application. Later on, we will use GANs, object detection models, facial recognition algorithm, machine learning algorithms, movement analysis and map matching algorithms.

We use the following algorithm to **divide the nearest cameras** evenly to the servers to avoid latency. Later on, we will divide the cameras not just based on their location but also based on the server’s specifications.

Get cameras count

Integer n= (floor) Divide cameras count by number of servers

For each server

Find nearest n cameras that are not attached to another server

Attach these cameras to server

We have used the following algorithm to **manage QRs:**

If Authoritative person is logged in

He adds members of the QR unit into the database

Adds QR unit into the database

Generates QR code containging the ID of the QR unit

QR unit opens the mobile application and presses login

QR unit scans the QR code and is logged in

If the authoritative person wants to log the QR unit out, he clicks on the force logout button next to the QR unit

Main server sends request to Firebase to add a force logout entry

As each QR unit is actively listening to the Firebase, the specific QR unit will be logged out.

For transforming pictures, we will use **cycleGAN.**

Input: Dataset of childhood pictures and dataset of adulthood pictures.

Preprocess images

Make a generator model

Make a discriminator model

Compile Models for transformation

Train the combined model

For location prediction, we will use **neural networks**

Input: People’s previous location

Make a model for matching people’s previous location to current location

Train the model till accurately determines current location

Test for new previous locations.

# Software requirements traceability matrix

Table 1 Requirements Traceability Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Req. Number** | **Ref. Item** | **Design Component** | **Component Items** |
| UC1.1 | Class Diagram | Camera | Add\_camera |
| UC1.2 | Class Diagram | Camera | Remove\_camera |
| UC1.3 | Class Diagram | Camera | Search\_cameras |
| UC1.4 | Class Diagram | Camera | Update\_info |
| UC1.5 | Class Diagram | Camera | View\_live\_footage |
| UC2.2 | Class Diagram | Server | Add\_server |
| UC2.3 | Class Diagram | Server | Remove\_Server |
| UC2.4 | Class Diagram | Server | Get\_connected\_cameras |
| UC2.5 | Class Diagram | Server | Add\_camera |
| UC2.1 | Class Diagram | Server | Auto\_assign\_camera |
| UC6.1 | Class Diagram | AuthoritativePerson | Get\_suspect\_track\_history |
| UC6.2 | Class Diagram | LocationTracker | Get\_probable\_path |
| UC6.3 | Class Diagram | LocationTracker | Predict\_next\_location |
| UC7.1 | Class Diagram | AuthoritativePerson | Get\_active\_mission |
| UC8.1 | Class Diagram | AuthoritativePerson | Login\_AP |
| UC8.2 | Class Diagram | AuthoritativePerson | Signup\_QRUnit |
| UC8.3 | Class Diagram | AuthoritativePerson | Signup\_admin |
| UC8.4 | Class Diagram | AuthoritativePerson | Signup\_authoritativePerson |
| UC8.5 | Class Diagram | QRUnit | Login\_QR |
| UC8.6 | Class Diagram | Admin | logout |
| UC8.7 | Class Diagram | AuthoritativePerson | Login\_QRUnit |
| UC8.8 | Class Diagram | AuthoritativePerson | Logout\_AP |
| UC8.8 | Class Diagram | AuthoritativePerson | Signup\_admin |
| UC9.1 | Class Diagram | QRUnit | Send\_message |
| UC9.2 | Class Diagram | QRUnit | call |
| UC9.3 | Class Diagram | QRUnit | ping |
| UC9.6 | Class Diagram | AuthoritativePerson | Get\_QRUnits |
| UC10.2 | Class Diagram | Suspect | Search\_suspect |
| UC10.3 | Class Diagram | Suspect | Add\_suspect |
| UC10.4 | Class Diagram | Suspect | Remove\_suspect |
| UC10.5 | Class Diagram | Suspect | Update\_info |
| UC10.6 | Class Diagram | AuthoritativePerson | Get\_suspect\_track\_history |

# Human interface design

Describe the functionality of the system from the user’s perspective. Explain how the user will be able  to use  your system to complete  all the  expected  features and  the  feedback  information that will be displayed for the user.

## Screen images

Display screenshots showing the interface from the user’s perspective. These can be hand-drawn, or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)

## 8.2 Screen objects and actions

A discussion of screen objects and actions associated with those objects

# References

* Designing UML: http://agilemodeling.com/artifacts/
* For guidance: Ian Sommerville – Software Engineering 9th Edition– Chapter 6
* For detailed system understanding: <https://medium.com/@madhawavidanapathirana/https-medium-com-madhawavidanapathirana-real-time-human-detection-in-computer-vision-part-1-2acb851f4e55>