**Automated Video Surveillance System Using**

**Python and a Mobile App**

A Project Report

*Submitted By*

**Rishi Verma (11BCE0377)**

**Satvik Dhandhania (11BCE0431) Sahil Govel (11BCE0472)**

*in partial fulfillment for the award of the degree of*

**Bachelor of Technology**

*In*

**Computer Science and Engineering**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

****

**May 2015**

****

**School of Computing Science and Engineering**

**DECLARATION**

We hereby declare that the project entitled **“Automated Video Surveillance System Using Python and a Mobile App”** submitted by us to the School of Computing Science and Engineering, VIT University, Vellore in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out by us under the supervision of **Dr. Senthil Jayavel, Associate Professor and Asst. Director Academics (Systems).** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

**Rishi Verma (11BCE0377)**

**Sahil Govel (11BCE0472)**

**Satvik Dhandhania(11BCE0431)**



**School of Computing Science and Engineering**

**CERTIFICATE**

The project report entitled “**Automated Video** Surveillance **System Using Python and a Mobile App**” is prepared and submitted by **Rishi Verma (Register No: 11BCE0377), Sahil Govel (Register No: 11BCE0472)** and **Satvik Dhandhania (Register No: 11BCE0431)**.It has been found satisfactory in terms of scope, quality and presentation as partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** in **VIT University, India.**

**Internal Guide**

**Dr. Senthil Jayavel**

**Associate Professor**

**Internal Examiner External Examiner**

**ACKNOWLEDGEMENT**

We are using this opportunity to express my gratitude to everyone who supported us throughout the course of this Engineering project. We are thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. We are sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project. A special gratitude we give to our Project Guide, Dr. Senthil Jayavel, for his continuous support and guidance.

We would also like to express our gratitude to Prof. Hari Seetha (Program Chair, B.Tech CSE) and Prof. Venkateswaran S (Dean) who provided us with the facilities being required and conductive conditions for our Engineering project.

Finally, we would like to express our sincere thanks to Vellore Institute of Technology, which has supported us with this project.

**Thank you,**

**Rishi Verma (11BCE0377)**

**Sahil Govel (11BCE0472)**

**Satvik Dhandhania (11BCE0431)**

**Table of  Contents**

Executive Summary.................................................................1

1. Introduction....................................................................2
   1. Objective................................................................2
   2. Motivation..............................................................2
   3. Background.............................................................2

2. Project Description  and  Goals..............................................3

3. Design Approach  and  Detail.................................................4

3.1 Viola  jones  face  detection..............................................4

3.2  Google Cloud Messaging..............................................10

4. Technical Specification......................................................13

4.1 Software Specification.................................................13

4.2 Google Cloud Messaging Components and Message format..14

4.3  Codes and Standards..................................................18

4.4 Constraints...............................................................19

4.5  Tradeoffs.................................................................19

5. Schedule, Tasks and Milestones...............................................20

6. Project Demonstration........................................................22

7. Market and Cost Analysis.....................................................24

8. Summary.........................................................................25

9. References......................................................................26

Appendix  ……......................................................................27

**List of Tables**

|  |  |
| --- | --- |
| **Title** | **Page** |
| Table 1 ­ GCM Specifications | 14 |
| Table 2 ­ Attributes of HTTP or XMPP messages | 15-18 |
| Table 3 ­ Cost Analysis | 24 |

**List of Figures**

|  |  |
| --- | --- |
| **Title** | **Page** |
| Figure 1 ­ Block Diagram | 4 |
| Figure 2 ­ HAAR Classifiers | 5 |
| Figure 3 ­ Attention Cascading for fast rejection | 6 |
| Figure 4A,4B - SIFT | 8 |
| Figure 5 ­ Output | 9 |
| Figure 6 - Output | 10 |
| Figure 7 – GCM Architecture | 11 |
| Figure 8 – Registration Phase | 12 |
| Figure 9 – Sending Message | 12 |
| Figure 10 – Notification Received | 22 |
| Figure 11 – Android Application Interface | 23 |
| Figure 12 – Face detection code | 23 |
| Appendix A – Gantt Chart | 27 |
| Appendix B – Flow Diagram | 28 |

**List of Abbreviations**

|  |  |
| --- | --- |
| **Abbreviation** | **Expansion** |
| SaaS | Software­as­a­Service |
| PaaS | Platform­as­a­Service |
| GCM | Google cloud messaging service |
| HTTP(S) | Hyper Text Transfer Protocol (Secure) |
| JSON | JavaScript Object Notation |
| SMS | Short Message Service |
| GSM | Global System for Mobile Communications |
| ID | Identity Document |
| API | Application Programming Interface |
| SIFT | Scale-Invariant Feature Transform |

**Executive Summary**

In the world today security and surveillance is a multibillion dollar industry. Video surveillance is increasing significance approach as organizations seek to safe guard physical and capital assets. At the same time, the necessity to observe more people, places, and things coupled with a desire to pull out more useful information from video data is motivating new demands for scalability, capabilities, and capacity. These demands are exceeding the facilities of traditional analog video surveillance approaches. Providentially, digital video surveillance solutions derived from different data mining techniques are providing new ways of collecting, analyzing, and recording colossal amounts of video data.

The scenes obtained from a surveillance video are usually with low resolution. Most of the scenes captured by a static camera are with minimal change of background. Objects in the outdoor surveillance are often detected in far field. Most existing digital video surveillance systems rely on human observers for detecting specific activities in a real-time video scene. However, there are limitations in the human capability to monitor simultaneous events in surveillance displays.

The project will provide a portable mobile app which can detect intrusion in your home or office by alerting activity of anyone in the video feed. This will give real-time notifications and will raise any red flags when there are anomalies in front of camera.

1. **Introduction**
   1. **Objective**

The objective is to monitor the behavior, activities, or other changing information, usually of people for the purpose of influencing, managing, directing, or protecting them. In this study, we focus on detecting humans and do not consider recognition of their complex activities. Human detection is a difficult task from a machine vision perspective as it is influenced by a wide variety of factors and methods to store images in the computer. There are two main purposes to a video surveillance system. The first and most important one is to deter crime. The second is to help catch criminals when a crime has been committed.

* 1. **Motivation**

The primary purpose of this project is to free the owner of the extensive task of manually reviewing the video to identify the person who is present in front of the Video Surveillance System. Thus, the users of this system will be notified through the mobile app which will receive push notifications from the Video Surveillance system and display it to the owner. For each person it detects, it will give the name of the person if and only if he/she exists in the predefined database, otherwise it will pass the message that an Unknown person has been detected. Thus the user will be informed remotely about the people present in front of the surveillance system.

* 1. **Background**

For an intelligent video surveillance system, the detection of a human being is important for abnormal event detection, person identification and tracking etc. A system, which needs to be programmed according to the location it is to be deployed in, would require lots of initial overheads while installing it. This overhead includes enumeration of the kind of activities which would happen in this area and then coming up with a model which accurately captures routine activities and flags non routine ones. Clearly, this overhead is large and makes a programmed approach unsuitable for large-scale deployment. Hence there is a need for unsupervised video surveillance system, which is able to learn routine activities on its own data. A system with self-learning ability would be easy to deploy and would make it possible to have large scale monitoring. The system includes the following points:

* + - **Person tracking and identification:** A person in a visual surveillance system can be identified using face recognition and gait recognition techniques. The detection and tracking of multiple people in cluttered scenes at public places is difficult due to a partial or full occlusion problem for either a short or long period of time.
    - **Abnormal event detection:** The most obvious application of detecting humans in a surveillance video is to early detect an event that is not normal. Abnormal events are classified as single-person loitering, multiple-person interactions (e.g. fighting and personal attacks) and person-facility/location interactions (e.g. object left behind and trespassing). Detecting sudden changes and motion variations in the points of interest and recognizing human action could be done by constructing a motion similarity matrix or adopting a probabilistic method. Methods based on probability statistics use the minimum change of time and space measure to model the method of probability.
    - **Notifying the system administrator/owner:** Once an unknown person is detected, the intelligent system should send a message to the administrator that an unknown person has been detected or the name of the person from the list of safe people.

1. **Project Description and Goals**

There are two main purposes to a video surveillance system. The first and most important one is to deter crime. The second is to help catch criminals when a crime has been committed. Some of the ways a video surveillance system accomplishes both these goals is by doing the following:

* 1. Monitoring stores and stock.
  2. Providing a visible presence or warning that video cameras are used in the store
  3. Recording any intrusion.
  4. Allowing people to monitor the cameras and see what is happening at any time of the day.
  5. Providing an identification method, by which people can be screened before entering.
  6. Allowing security personnel to check who is in a building at any time.

There are two major computing aspects to the system:

1. **The Viola/jones face detector**- This is a machine learning approach for visual object detection, which is capable of processing images extremely rapidly and achieving high detection rates.
2. **Android app**- This application is used to receive the push notifications from the Surveillance System remotely via the Google Cloud Messaging service.
3. **Design Approach and Details**
   1. **Design Approach**

As developers of the project the first task is to identify the various components of the product. The product is primarily an open CV code which generates results of face detection and object tracking. We then send this as a message to the mobile app through push notifications.

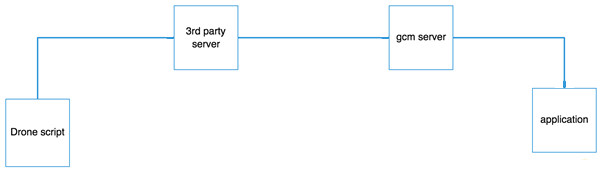
****

Figure 1 Block Diagram

**The Viola/Jones face detector**

This method is a machine learning approach for visual object detection, which is capable of processing images extremely rapidly and achieving high detection rates. This work is distinguished by three key contributions:

* + 1. The first is the introduction of a new image representation called the “Integral Image” which allows the features used by our detector to be computed very quickly.
    2. The second is a learning algorithm, based on AdaBoost, which selects a small number of critical visual features from a larger set and yields extremely efficient classifiers.
    3. The third contribution is a method for combining increasingly more complex classifiers in a “cascade” which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions.

**AdaBoost**

In our system a variant of AdaBoost is used both to select a small set of features and train the classifier. In its original form, the AdaBoost learning algorithm is used to boost the classification performance of a simple (sometimes called weak) learning algorithm.

There are a number of formal guarantees provided by the AdaBoost learning procedure. Freund and Schapiro proved that the training error of the strong classifier approaches zero exponentially in the number of rounds. More importantly a number of results were later proved about generalization performance.

The key insight is that generalization performance is related to the margin of the examples and AdaBoost achieves large margins rapidly. Recall that there are over 180,000 rectangle features associated with each image sub-window, a number far larger than the number of pixels. Even though each feature can be computed very efficiently, computing the complete set is prohibitively expensive. Our method, which is borne out by experiment, is that a very small number of these features can be combined to form an effective classifier.

The main challenge is to find these features. In support of this goal, the weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples (this is similar to the approach in the domain of image database Retrieval). For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples is misclassified.

Boosting is a classification scheme that works by combining weak learners into a more accurate ensemble classifier

Weak learner: classifier with accuracy that need be only better than chance

We can define weak learners based on rectangle features:

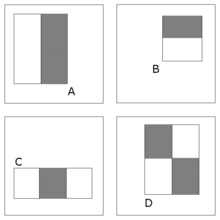


Figure 2 HAAR Classifier

(It is similar to HAAR wavelets)

The process of boosting:

Initially, give equal weight to each training example

* + Iterative training procedure.
  + Find best weak learner for current weighted training set.
  + Raise the weights of training examples misclassified by current weak learner.
  + Compute final classifier as linear combination of all weak learners (weight of each learner is related to its accuracy). Fig below shows how the process works, similar to HAAR classifiers.

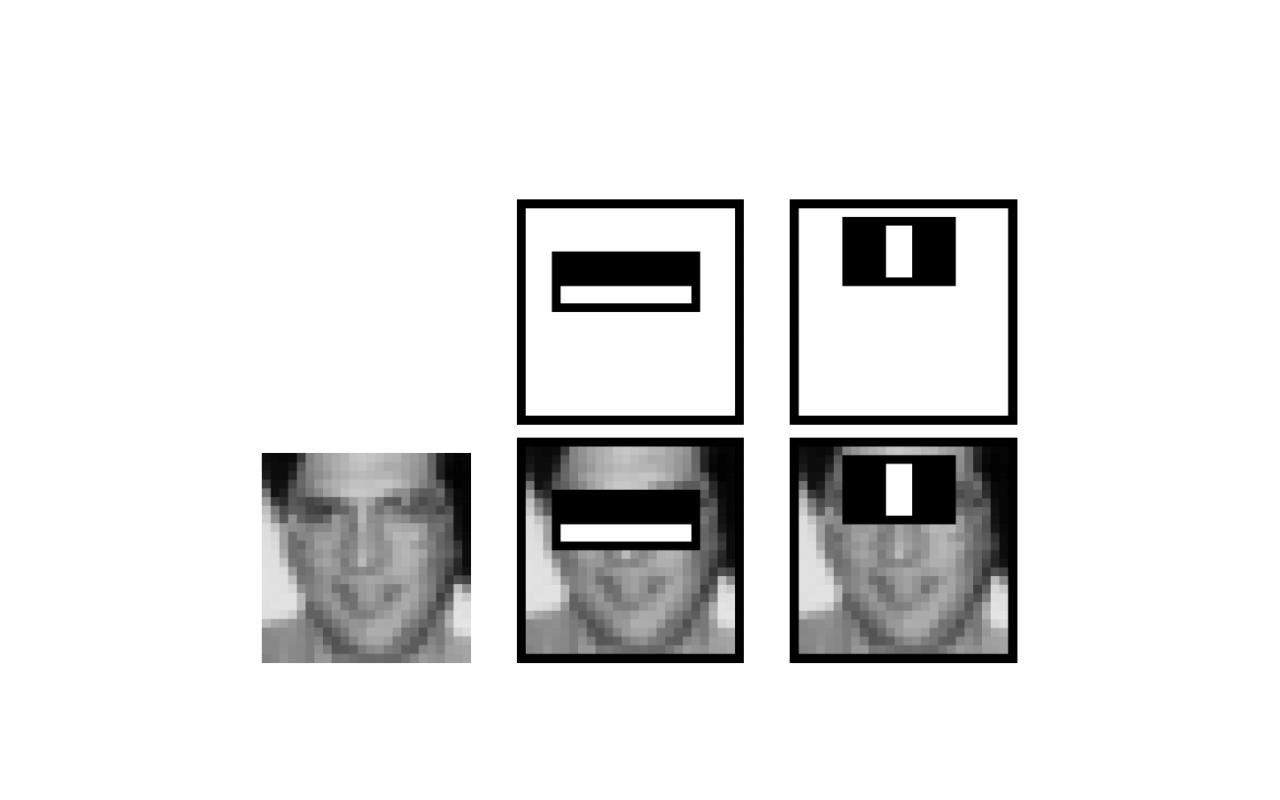


Figure 3 Attention Cascading for fast rejection of non-face windows

Attention cascade for fast rejection of non-face windows

The steps involved in the process are:

a) We start with simple classifiers, which reject many of the negative sub-windows while detecting almost all positive sub-windows.

b) Positive results from the first classifier trigger the evaluation of a second (more complex) classifier, and so on.

c) A negative outcome at any point leads to the immediate rejection of the sub-window.

**Training the cascade**

a) Adjust weak learner threshold to minimize false negatives (as opposed to total classification error).

b) Each classifier trained on false positives of previous stages.

c) A single-feature classifier achieves 100% detection rate and about 50% false positive rate.

d) A five-feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative) • A 20-feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative).

**Key points detection (SIFT) (Scale-Invariant Feature Transform)**

For any object in an image, interesting points on the object can be extracted to provide a "feature description" of the object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. To perform reliable recognition, it is important that the features extracted from the training image be detectable even under changes in image scale, noise and illumination. Such points usually lie on high-contrast regions of the image, such as object edges.

Another important characteristic of these features is that the relative positions between them in the original scene shouldn't change from one image to another. For example, if only the four corners of a door were used as features, they would work regardless of the door's position; but if points in the frame were also used, the recognition would fail if the door is opened or closed. Similarly, features located in articulated or flexible objects would typically not work if any change in their internal geometry happens between two images in the set being processed. However, in practice SIFT detects and uses a much larger number of features from the images, which reduces the contribution of the errors caused by these local variations in the average error of all feature matching errors.

SIFT can robustly identify objects even among clutter and under partial occlusion, because the SIFT feature descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes. This section summarizes Lowe's object recognition method and mentions a few competing techniques available for object recognition under clutter and partial occlusion.

There are mainly 4 steps involved in SIFT, they are:

1. **Scale-space Extrema Detection**

To detect larger corners we need larger windows. For this, scale-space filtering is used. In it, Laplacian of Gaussian is found for the image with various values. LoG acts as a blob detector, which detects blobs in various sizes due to change in. In short, acts as a scaling parameter. For example, in the above image, Gaussian kernel with low gives high value for small corner while Gaussian kernel with high fits well for larger corner. So, we can find the local maxima across the scale and space which gives us a list of values which means there is a potential key point at (x,y) at scale. But this LoG is a little costly, so SIFT algorithm uses Difference of Gaussians which is an approximation of LoG. Difference of Gaussian is obtained as the difference of Gaussian blurring of an image with two different, let it be and this process is done for different octaves of the image in Gaussian Pyramid. It is represented in below image:

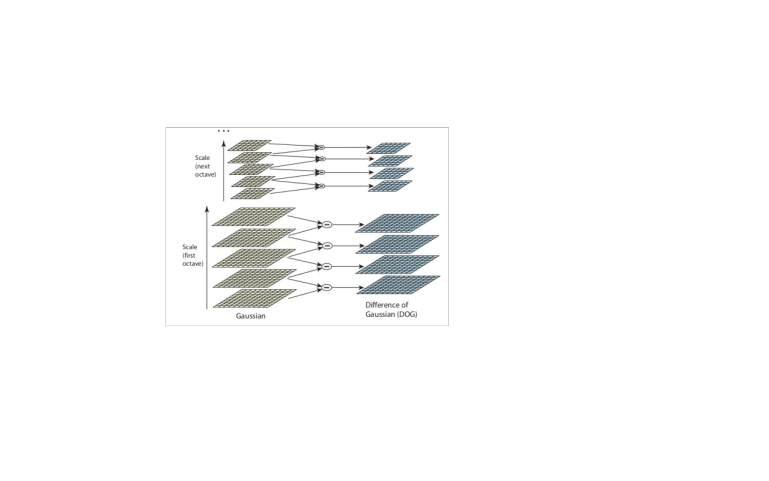


Figure 4A SIFT

Once this DoG is found, images are searched for local extrema over scale and space. For e.g., one pixel in an image is compared with its 8 neighbors as well as 9 pixels in next scale and 9 pixels in previous scales. If it is a local extrema, it is a potential key point. It basically means that key point is best represented in that scale. It is shown in below image:

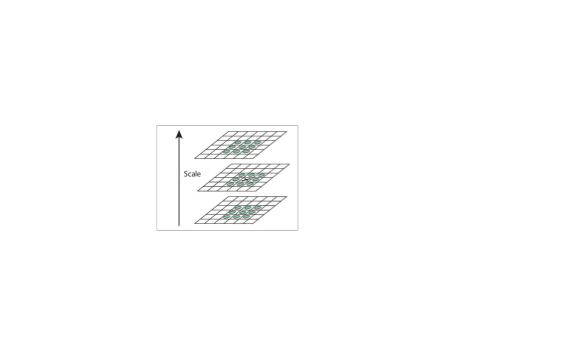


Figure 4B SIFT

**2. Keypoint Localization**

Once potential keypoints locations are found, they have to be refined to get more accurate results. They used Taylor series expansion of scale space to get more accurate location of extrema, and if the intensity at this extrema is less than a threshold value (0.03 as per the paper), it is rejected. This threshold is called contrast Threshold in OpenCV DoG has higher response for edges, so edges also need to be removed. For this, a concept similar to Harris corner detector is used. They used a 2x2 Hessian matrix (H) to compute the principal curvature. We know from Harris corner detector that for edges, one Eigen value is larger than the other. If this ratio is greater than a threshold, called edge Threshold in OpenCV, that keypoint is discarded. It is given as 10 in paper. So it eliminates any low-contrast keypoints and edge keypoints and what remain are strong interest points.

Now an orientation is assigned to each keypoint to achieve invariance to image rotation. A neighborhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region. An orientation histogram with 36 bins covering 360 degrees is created. (It is weighted by gradient magnitude and Gaussian- weighted circular window with equal to 1.5 times the scale of keypoint. The highest peak in the histogram is taken and any peak above 80% of it is also considered to calculate the orientation. It creates keypoints with same location and scale, but different directions. It contributes to stability of matching.

**3. Keypoint Descriptor**

Now keypoint descriptor is created. A 16x16 neighborhood around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 sizes. For each sub-block, 8-bin orientation histogram is created. So a total of 128 bin values are available. It is represented as a vector to form keypoint descriptor. In addition to this, several measures are taken to achieve robustness against illumination changes, rotation etc.

**4. Keypoint Matching**

Keypoints between two images are matched by identifying their nearest neighbors. But in some cases, the second closest-match may be very near to the first. It may happen due to noise or some other reasons. In that case, ratio of closest-distance to second-closest distance is taken. If it is greater than 0.8, they are rejected. It eliminators around 90% of false matches while discards only 5% correct matches.

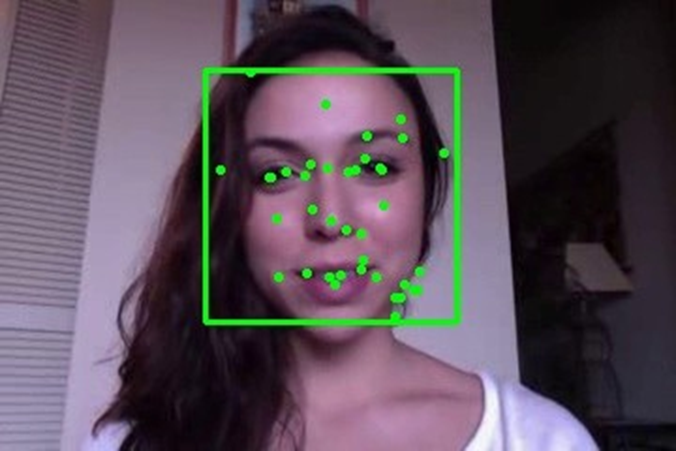
****

Figure 5 Output

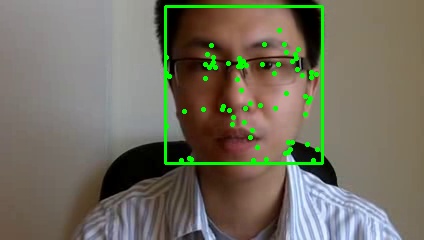
****

Figure 6 Output

**Google Cloud Messaging**



Google Cloud Messaging (GCM) is a free service that enables developers to send downstream messages (from servers to GCM-enabled client apps), and upstream messages (from the GCM-enabled client apps to servers). This could be a lightweight message telling the client app that there is new data to be fetched from the server (for instance, a "new email" notification informing the app that it is out of sync with the back end), or it could be a message containing up to 4kb of payload data (so apps like instant messaging can consume the message directly). The GCM service handles all aspects of queueing of messages and delivery to and from the target client app.

**Architectural Overview**

A GCM implementation includes a Google-provided connection server, a 3rd-party app server that interacts with the connection server, and a GCM-enabled client app. For example, this diagram shows GCM communicating with a client app on an Android device:

Figure 7 GCM Architecture

This is how these components interact:

* Google-provided **GCM Connection Servers** take messages from a 3rd-party app Server and send these messages to a GCM-enabled client app (the "client app"). Currently Google provides connection servers for HTTP and XMPP.
* The **3rd-Party App Server** is a component that we implement to work with our chosen GCM connection servers. The App server sends messages to the GCM connection server which then enqueues and stores the message, and finally sends it to the client app.
* The **Client App** is a GCM-enabled client app. To receive GCM messages, this app must register with GCM and get a registration ID. If we are using the XMPP (CCS) Connection server, the client app can send "upstream" messages back to the 3rd-party app server also.

**Lifecycle Flow**

**Client Side (Mobile Application)**

* **Register to enable GCM**. A client app registers to receive messages.
* **Send and receive downstream messages.**

1. **Receive a message**. A client app receives a message from a GCM server.
2. **Send and receive upstream messages.** This feature is only available if you're using the XMPP Cloud Connection Server (CCS).

* **Send a message**. A client app sends messages to the 3rd-party app server:

1. On the device, the client app sends messages to XMPP (CCS).
2. XMPP (CCS) enqueues and stores the message if the server is disconnected.
3. When the 3rd-party app server is re-connected, XMPP (CCS) sends the message to the 3rd-party app server.

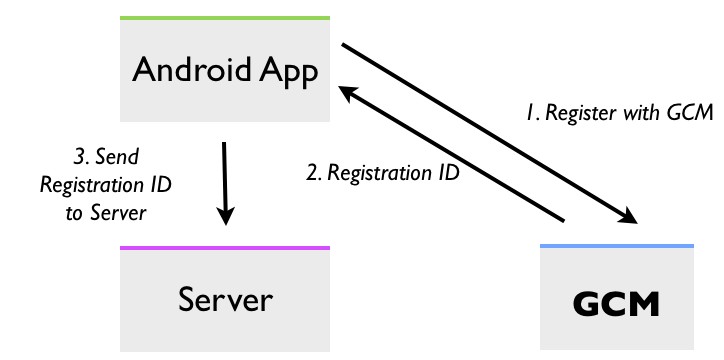
****

Figure 8 Registration Phase

**3rd Party Server**

**Sending a message to the client application**

1. The 3rd-party app server sends a message to GCM connection servers.
2. The GCM connection server enqueues and stores the message if the Device is offline.
3. When the device is online, the GCM connection server sends the Message to the device.
4. On the device, the client app receives the message according to the platform.

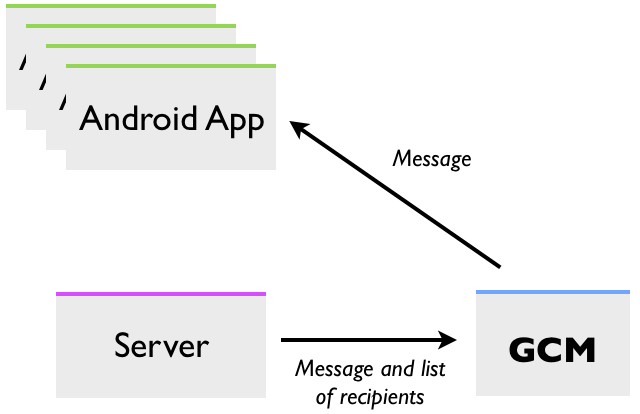
****

Figure 9 Sending Messages

**Receive a message.** A 3rd-party app server receives a message from XMPP (CCS) and then does the following:

1. Parses the message header to verify client app sender information.
2. Sends "ack" to GCM XMPP connection server to acknowledge receiving the message.
3. Optionally parses the message payload, as defined by the client app.

1. **Technical Specifications**
   1. **Software Specification**

**Android Studio**

Android Studio is the official IDE for Android application development, based on IntelliJ IDEA. We use this software to develop our Android application which receives the message from the Google Cloud Messaging Server. Android Studio offers:

* Flexible Gradle-based build system
* Build variants and multiple APK file generation
* Code templates to help you build common app features
* Rich layout editor with support for drag and drop theme editing
* lint tools to catch performance, usability, version compatibility, and other problems
* ProGuard and app-signing capabilities
* Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

**Google App Engine**

Google App Engine (often referred to as GAE or simply App Engine) is a platform as a service (PaaS) cloud computing platform for developing and hosting web applications in Google-managed data centers. Applications are sandboxed and run across multiple servers. App Engine offers automatic scaling for web applications—as the number of requests increases for an application, App Engine automatically allocates more resources for the web application to handle the additional demand. Google App Engine is free up to a certain level of consumed resources. Fees are charged for additional storage, bandwidth, or instance hours required by the application. We use the app engine to host our 3rd Party Server which receives the message from the python application and forwards it to Google Cloud Messaging Server along with the registration ID’s of all the registered devices.

**OpenCV**

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel Russia research center in Nizhny Novgorod. It is free for use under the open-source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself. We make use of OpenCV libraries for video processing of the input video to identify people.

* 1. **Google Cloud Messaging Components and Message format**

The cloud messaging systems needs certain components to function properly. The components work together to provide the whole service. Any component failing to execute properly will fail the whole system. The overall system components have been described and stated in Table 4.1.

|  |  |
| --- | --- |
| **Components** | |
| **GCM Connection Servers** | The Google-provided servers involved in sending messages between the  3rd-party app server and the client app. |
| **Client App** | A GCM-enabled client app that communicates with a 3rd-party app server. |
| **3rd-party App**  **Server** | An app server that you write as part of implementing GCM. The 3rd-party app server sends data to a client app via the GCM connection server. |
| **Credentials** | |
| **Sender ID** | A project number you acquire from the API console. The sender ID is used in the registration process to identify a 3rd-party app server that is permitted to send messages to the client app. |
| **Sender Auth**  **Token** | An API key that is saved on the 3rd-party app server that gives the app server authorized access to Google services. The API key is included in the header of POST requests. |
| **Application ID** | The client app that is registering to receive messages. How this is implemented is platform-dependent. For example, an Android app is identified by the package  name from the manifest. This ensures that the messages are targeted to the  correct Android app. |
| **Registration ID** | An ID issued by the GCM servers to the client app that allows it to receive messages. Note that registration IDs must be kept secret. |

Table 1 GCM Specifications

**Downstream HTTP or XMPP messages (JSON)**

The following table lists the targets, options, and payload for HTTP or XMPP JSON messages.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Parameter** | **Protocol** | | **Usage** | | | **Description** |
|  | **Targets** | | | | | | |
|  | To | XMPP | | Required,  string | | | This parameter specifies the recipient of a message.  The value must be a registration ID  /notification key.  This parameter is used in XMPP in place of registration\_ids or notification\_key in HTTP. |
|  | registration\_ids | HTTP | | Required  if  notification\_key  not  present,  string  array | | | This parameter specifies the list of devices (registration IDs) receiving the message. It must contain at least  1 and at most 1000 registration IDs.  Multicast messages (sending to more than 1 registration IDs) are allowed using HTTP JSON format only.  This parameter or notification\_key is used in HTTP in place of to in XMPP. |
|  | notification\_key | HTTP | | Required if registration\_ids not present, string | | | This parameter specifies the mapping of a single user to multiple registration IDs associated with that user.  This allows a 3rd-party app server to send a single message to multiple app instances (typically on multiple devices) owned by a single user.  A 3rd-party app server can use notification\_key as the target for a message instead of an individual registration ID (or array of registration IDs). The maximum number of members allowed for a notification\_key is 20.This parameter or registration\_ids is used in HTTP in place of to in XMPP. |
|  | **Options** | | | | | | |
|  | message\_id | | XMPP | | Required,  string | This parameter uniquely identifies a message in an XMPP connection. | |
|  | collapse\_key | | HTTP,  XMPP | | Optional,  string | This parameters identifies a group of messages (e.g., with collapse\_key: "Updates Available") that can be collapsed, so that only the last message gets sent when delivery can be resumed. This is intended to avoid sending too many of the same messages when the device comes back online or becomes active.  Note that there is no guarantee of the order in which messages get sent.  Messages with collapse key are also called send-to-sync messages messages.  Note: A maximum of 4 different collapse keys is allowed at any given time. This means a GCM connection server can simultaneously store 4 different send-to-sync messages per client app. | |
|  | delay\_while\_idle | | HTTP,  XMPP | | Optional,  JSON Boolean | When this parameter is set to true, it indicates that the message should not be sent until the device becomes active.  The default value is false. | |
|  | time\_to\_live | | HTTP,  XMPP | | Optional,  JSON number | This parameter specifies how long (in seconds) the message should be kept in GCM storage if the device is offline. The maximum time to live supported is 4 weeks.  The default value is 4 weeks. | |
|  | delivery\_receipt\_  requested | | XMPP | | Optional,   JSON boolean | This parameter lets 3rd-party app server request confirmation of message delivery.  When this parameter is set to true, CCS sends a delivery receipt when the device confirms that it received the message.  The default value is false. | |
|  | restricted\_package\_  name | | HTTP | | Optional,  string | This parameter specifies the package name of the application where the registration IDs must match in order to receive the message. | |
|  | dry\_run | | HTTP | | Optional,   JSON boolean | This parameter, when set to true, allows developers to test a request without actually sending a message.  The default value is false. | |
|  | **Payload** | | | | | | |
|  | Data | | HTTP,  XMPP | | Optional,   JSON  object | This parameter specifies the key- value pairs of the message's payload. There is no limit on the number of key-value pairs, but there is a total message size limit of 4kb.  For instance, in Android,  data:{"score":"3x1"} would  result in an intent extra named score  with the string value 3x1.  The key should not be a reserved word (from or any word starting with google). It is also not recommended to use words defined in this table (such as collapse\_key) because that could yield unpredictable outcomes.  Values in string types are recommended. You have to convert values in objects or other non-string data types (e.g., integers or booleans) to string. | |

Table 2 Attributes of HTTP or XMPP Messages (JSON)

**Request received by GCM Server**

curl --header "Authorization: key=AIzaSyD6ktqBEKOqAp4jvYV8r9eth6hhP0KaQE4" --header Content-Type:"application/json" https://android.googleapis.com/gcm/send -d "{\"registration\_ids\":[\" APA91bF4l4XEDbN6ywIYA9aEPxS\_V2JY4f\_LNPMEUsqF1JMDVC4vawTHFKxI-XHRCU6aifpnjSXZD6aA7OY7rLkH4Ml9apc8kEgkUZ-bVaKrMRA8ZfHsw3QDBLGKiJs9mLZW9EKqAeFaE6ieUnuCU114lAh6Ys25HA \"], \"data\":{\"name\":\"Satvik\"}}"

* 1. **Codes and Standards**

1. UMTS, WCDMA ­ HSPA/HSPA+: 3G data standards
2. USB: Needed to interface the the smartphone to the computer
3. IEEE 802.11: WiFi standard
4. RFC 2616: HTTP/1.1 standard
5. UHF radio waves in the ISM band from 2.4 to 2.485 GHz
6. NIEM 3.0 for XML encoding of the standard. Currently NIEM 2.1 is used for ANSI/NIST­ITL schema.
   1. **Constraints**

The following have been identified as the constraints of the product:

1. **Economic Constraint:** The system will require the deployment of the video surveillance system at every entrance.
2. **Social Constraint**: The usage of such a system can be deemed as violation of privacy in public areas.
3. **Sustainability Constraint:** The usage of this system has to be sustained for large campuses and needs to be scaled accordingly; the sustainability of such an expansive system needs to be tested.
4. **Logistical Constraint:** The system cannot take large amount of data’s and it requires active internet connection all the time.
5. **Environment Constraint:** The system may not function properly in certain environments like in dim lights or weak internet connection zones.
6. **Delivery Constraint:** The application completely depends on the Google Cloud Messaging platform which may add delay to send the notifications to users since it is a free and open service.
   1. **Tradeoffs**

**Ease of Installation vs Ease of Use:**

The installation of the application for an amateur user is difficult due to the fact that many libraries of OpenCV and HTTP requests need to be installed by the user for setting up the surveillance system. The application also needs a runtime environment so that it can send messages to the 3rd Party App server. This makes the installation phase difficult. However, once the setup phase is complete, the application is easy to use. On the other hand, the android application automatically handles the registration process with the GCM server which makes the android side to be easily installed.

**Smartphone vs GSM:**

The use of smartphone app makes the entire system very cost effective and the fact that everyone can now afford a smartphone it turns out to be easy to go for a web based messaging service as compared to the other alternatives like GSM Smart Messaging System. The SMS service also may create false alarms as the message may be a service message from the network provider. Thus having push notifications helps us categorize the messages to be app specific and thus only the relevant messages would be received by the application. Also, the cost of the system would increase as each SMS would be charged by the carrier.

1. **Schedule, Tasks and Milestones**

**Schedule**

The development process for the Surveillance System follows the conventional waterfall model for software engineering. The whole process has been divided into the following:

1. Project Management
   1. Project Initiation
   2. Project Planning
   3. Feasibility Analysis
2. Face Detection Algorithms Research and Implementation
3. Implementation of Android Application
4. Implementation of Servers
5. Linking of Individual Components to get the whole system
6. Testing
7. Documentation of results

The scheduled tasks have been successfully implemented and tested. The various components like python application (Tracking and Viola Jones for face detection), GCM connection server, 3rd Party Application Server and Android application have been interlinked and tested to be working as a Surveillance system.

**Tasks**

The different tasks associated with the requirements analysis and design phase of the system have been enumerated below:

1. **Background and Study of existing techniques and methodologies -** This task involved the study of various existing methods to tackle the task and looked for the most suitable solution for our problem which included study of various algorithms and their efficiencies, libraries that could be used and programming language that would be suitable for deployment. It also provided the whole architecture for the deployment of the whole system. The placement of servers and methods to interconnect the various servers to provide the push notification message.
2. **Installing and setting up all the OpenCV libraries and functions –** Once all the technologies and methods were decided upon, the next step was to install all the libraries and test them on the system and check whether it would work well for our problem or not.
3. **Designing the mobile App on Android Studio –** This task focused on installing Android studio and learning android programming and then finally designing the User Interface of the client application.
4. **Development of program that detects the face in a feed –** Once the libraries were installed, the next task was to detect faces in the input video feed. This was performed using Keypoint Localization strategy.
5. **Development of the 3rd Party Application Server on the Google App Engine–** This task involved the deployment of the 3rd party app server that handles registration and identification phases of the system. The registration phase accepts the Client apps request where the client app passes the registration ID it received from the GCM connection server. The 3rd party server stores this registration ID in the datastore of the app engine and thus maintains the list of registered users. Next, it handles the request of the python app where it receives the message and then passes it to the GCM server along with the registration ID’s of the all the registered devices.
6. **Interlinking of the 3rd Party Application Server to the Google Cloud Messaging Server –** In this task a Google API project is created in the Google Developers Console. We get our project number which we use while registration of the app to the server. Next, we enable the GCM service by going to APIs & auth and turn the Google Cloud Messaging for Android toggle to ON. Next, we obtain the API key by going to credentials and create new key in Public API access and create a new server key. This API key is used by the 3rd party application server to forward messages to GCM connection server.
7. **Configuring the Device to be able to receive messages and display them in correct format –** In this task the android application we receive the bundle which contains the message in JSON format. We parse it to extract our message and then add the current time to it. Finally, using the broadcast receiver the notification is generated and the user is notified about the message. Also, clicking on the notification creates an Activity and loads the application which displays past along with current received messages with timestamps.

**Milestones**

There are several major milestones as well as sever smaller tasks that must be achieved in order to reach the milestones. Tasks will be split up among group members according to each member’s level of expertise or comfort. Each task will have a leader who is responsible for completion of that task. However, the other group members are expected to provide assistance if needed. This way, an engineer can concentrate on a task but still get help if needed. Also, each tasks respective leader is held accountable if the task fails. A Gantt chart outlining important tasks and the schedule can be seen in the Appendix.

The milestones are:

* Selecting an algorithm for the computer vision face detection
* Designing the mobile App on Android Studio.
* Development of program that detects the face in a feed.
* Development of the 3rd Party Application Server on the Google App Engine
* Interlinking of the 3rd Party Application Server to the Google Cloud Messaging Server
* Testing the application for various test cases.

1. **Project Demonstration**

The project demonstration will start from the implementation and showing of the result of the desktop application, which will run a background script and starts detecting the faces in the video feed. This process will be followed by the push notification on the users mobile phone which will update the user about the activities in front of the camera even if the user is not nearby his/her system.

The project demonstration is divided into two parts:

* 1. **Facial recognition software**: in this part we are running our facial recognition algorithms which will detect any faces that are present in the video feed, it will first identify the key points using SIFT algorithms and then by using viola/jones algorithms it will start detecting faces.
  2. **Android mobile app:** the demonstration of the client mobile application was demonstrated on an android mobile phone. The mobile app will receive notifications from the servers and it will show the notifications received from results running on the system.

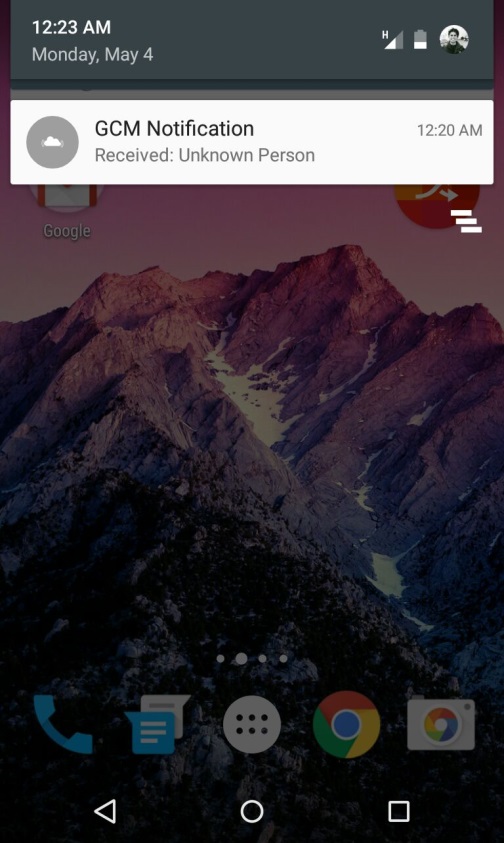
****

Figure 10 Notification Received

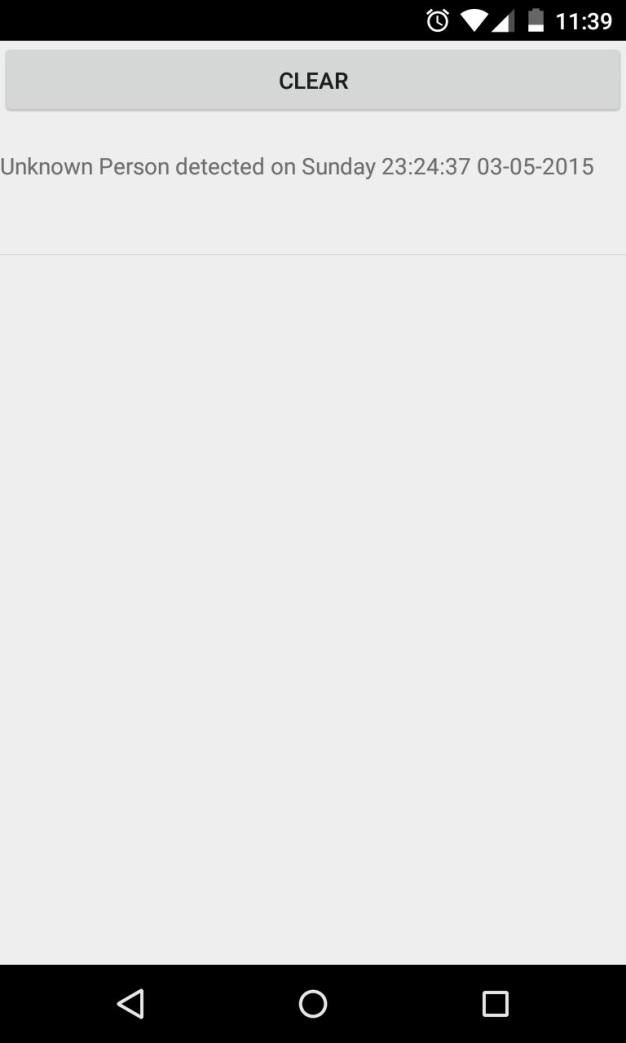
****

Figure 11 Android Application Interface

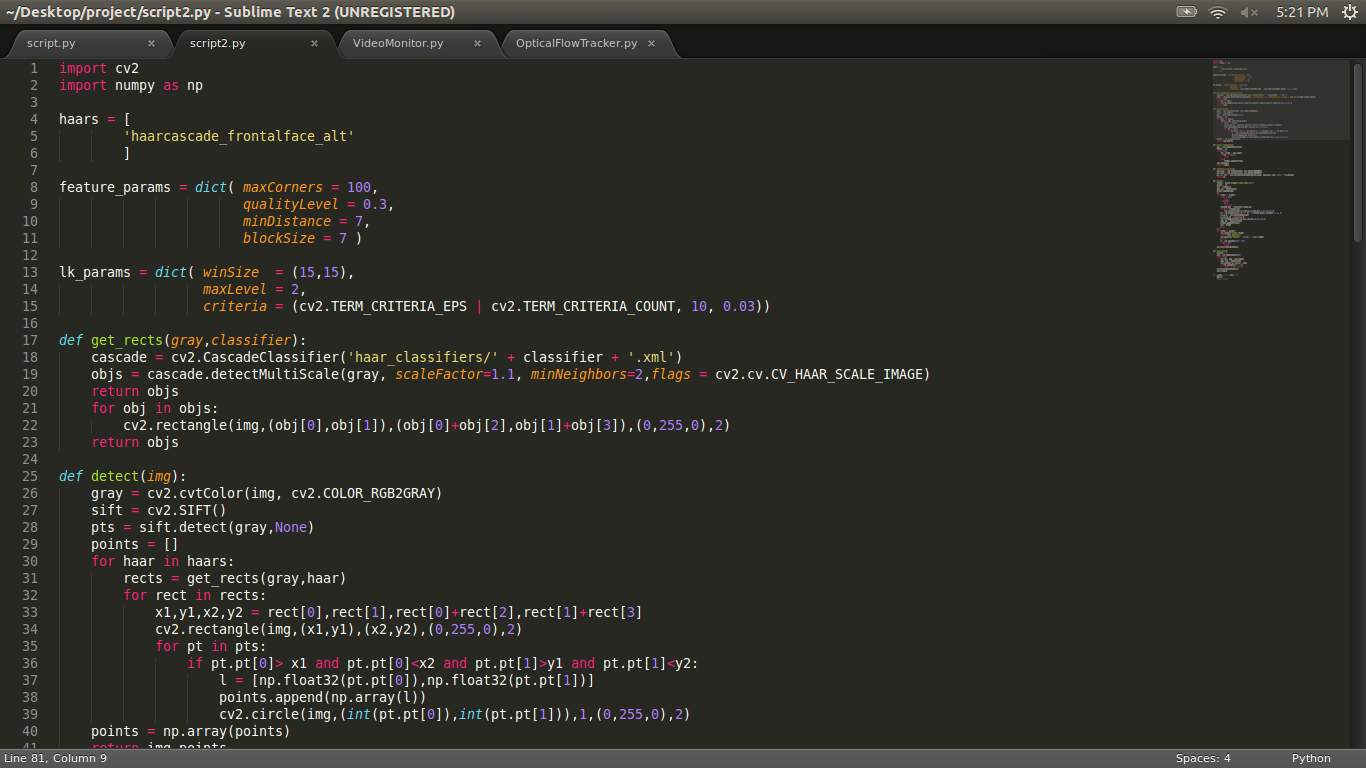
****

Figure 12 Face Detection Code

1. **Market and Cost Analysis**
   1. **Market Analysis**

The existing method used in most places is limited to a person who can see footage of camera to do the surveillance. This method is limited and susceptible to human error. This method also varies from place to place and is not very efficient and requires crowd sourcing.

Our product is not susceptible to manual error since the face detection is completely autonomous this helps the system to be efficient and fast. It is easy to use and portable and the application created specifically for the system can be used by any person with an android phone.

* 1. **Cost Analysis**

The cost analysis for the entire system boils down to the following part costs. The part costs are as described in the following table. The system will work with any smartphone with an active internet connection. However that will not be factored into the cost analysis because it is a common entity of use.

The online infrastructure cost for any institution should be nil as all open source resources have been used. When used in large contexts, in order to scale to serve more client endpoints then the online infrastructure must be expanded in a pay­per­usage manner which has also been included below:

|  |  |
| --- | --- |
| **Entity** | **Cost** |
| Basic smart phones | 50$ |
| Google Developers Console- One time Registration fee | 25$ |
| Online infrastructure cost Google App Engine (Only for usage past quota) | 0$­50$(15$ ­ DBaaS + 35$ ­ PaaS per month) |

Table 3 Cost Analysis

1. **Summary**

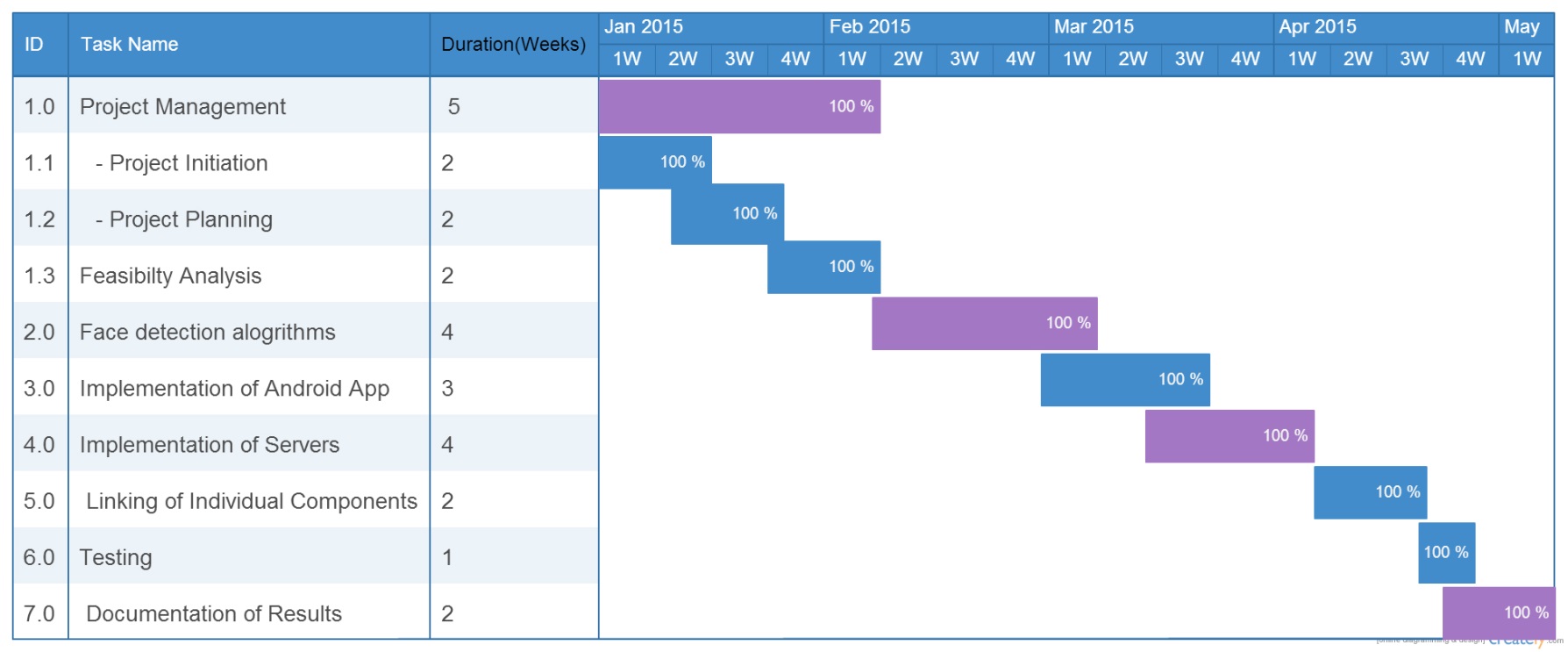
The project is aimed to provide a cost effective and easy to use video surveillance system which uses state of the art facial recognition algorithms to detect the faces in the video feed and alert the user on their smart phones using a mobile application. The objective is to monitor the behavior, activities, or other changing information, usually of people for the purpose of influencing, managing, directing, or protecting them. In this study, we focus on detecting humans and do not consider recognition of their complex activities. This overhead includes enumeration of the kind of activities which would happen in this area and then coming up with a model which accurately captures routine activities and flags non routine ones. Clearly, this overhead is large and makes a programmed approach unsuitable for large-scale deployment. Hence there is a need for unsupervised video surveillance system, which is able to learn routine activities on its own data. A system with self-learning ability would be easy to deploy and would make it possible to have large scale monitoring. The system includes the following points:

* + - Face detection and tracking and identification.
    - Abnormal event detection.
    - Registration of Android application with the Application Server and GCM connection server.
    - Interlinking of various components of the system to provide the discussed services.
    - Notifying the system administrator/owner on his/her android app via Push Notifications.

1. **References**
   1. M.Hunke and A. Waibel, “Face Locating And Tracking for Human Computer Interaction”,in Proc. Conf. Signals Systems and Computers, Nov 1994, vol2 pp 1277- 1281.
   2. Abbas El Gamal, “Introduction To Statistical Signal Processing”, Lecture Notes.
   3. D. Cahi and K.N. Ngan, “Face segmentation using skin color map”, IEEE Trans. On Circuit and Systems for Video Technology, vol 9, no 4, pp. 551-564, Jun. 1999.
   4. io.js JavaScript Runtime and Documentation: <https://iojs.org/en/index.html>.
   5. Sirovich, L. and Kirby, M.: 1987, Low dimensional procedure for the characterization of human faces, J. Opt. Soc. Am. A 4, 519–524.
   6. Handbook of Face Recognition, S.Z. Li and A.K. Jain, eds. Springer, 2005.
   7. M. Turk and A. Pentland, "Eigenfaces for Recognition," *J. Cognitive Neuroscience,* vol. 3, no. 1, pp. 71-86, 1991.
   8. K. Etemad and R. Chellappa, "Discriminant Analysis for Recognition of Human Face Images," *J. Optical Soc. Am.,* vol. 14, pp. 1724-1733, 1997.
   9. J. Yang, D. Zhang, A.F. Frangi and J. Yang, "Two-dimensional PCA: A New Approach to Appearance-Based Face Representation and Recognition," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 26, no. 1, pp. 131-137, Jan 2004.
   10. L. Wiskott, J.-M. Fellous, N. Kuiger and C. von der Malsburg, "Face Recognition by Elastic Bunch Graph Matching," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, pp. 775-779, 1997.
   11. B. Heisele, P. Ho, J. Wu and T. Poggio, "Face Recognition: Component-Based versus Global Approaches," Compter Vision and Image Understanding, vol. 91, nos.1-2,pp.6-21,2003.
   12. T. Ahonen, A. Hadid and M. Pietikä,inen, "Face Recognition with Local Binary Patterns," Proc. Eighth European Conf. Computer Vision, pp. 469-481, 2004.

**Appendix**

**A- Gantt Chart**

****

**B- Flow Diagram**

