King Saud University
College of Computer and Information Sciences
Department of Information Technology

جامعة الملك سعود
King Saud University
كلية علوم الحاسب والمعلومات
قسم تقنية المعلومات


*Graduation Project Report*

# iGuard
# Mobile Security Guard System

Prepared by

Tahani Alghanem, 435201330
Lama Almansour, 435200675
Afrah Alsubaie, 435201047
Lamia Alshahrani, 435200552

Supervised by
Dr. Ghada Al-Hudhud

Graduation Project Report
Semester 2 AY 1438/1439 – 2017/2018

# Contents

## List of Figures

## List of Tables

# STUDENT DECLARATION

Student acknowledges that he has read the above policy pertaining to plagiarism and understands that the penalty for such an act could result in disciplinary action.

**Project Title:**

*iGuard*: Mobile Security Guard System

**Academic Year:**

2017/2018 (Gregorian) -1438/1439 (Higri)

| | Student's Name | Student' s University ID Number | Student's Signature | Date |
|---|---|---|---|---|
| **1** | Afrah Alsubaie | 435201047 | | |
| **2** | Lama Almansour | 435200675 | | |
| **3** | Lamia Alshahrani | 435200552 | | |
| **4** | Tahani Alghanem | 435201330 | | |

# ABSTRACT

Securing work, institutional, business, tourism or military facilities is highly demanded. Current security management systems rely on hiring teams of security guards supported by a set of surveillance cameras and monitoring devices. These systems verify the visited points by security guards in the building/facility and report the deviation of security check plans. Meanwhile, these systems require at least one guard should monitor the cameras displays in a control room. In addition, the currently available mobile security guard systems; for example Pancomp, monitors the security guards performance not the facilities. The proposed mobile security management solution; iGuard, provides a powerful and cost effective mobile system to ensure running all security operations with less effort and more awareness. The proposed iGuard implements point to point (P2P) or point to multiple points casting of images and videos from the camera to Google glass worn by the security guard(s). This is quite important in order to overcome the problem of Google glass battery that might be of so that other security guard would receive the alarm and live stream.

**iGuard** integrates different technologies to include mobile phone, surveillance camera and Google glass and PIR sensor. The sensory data and camera records are sent to the main control room monitors. The system should have a look up table that contains normal out of working-hours; and accordingly any abnormal sensory data indicates immediate camera records check via the Google glass. Developing iGuard system requires availability of infrastructure to detect motion, in addition to a camera in the workplace or offices. These devices will be switched on to sense the environment during out of working hours. The normal data during out working-hours will be detected. Hence, if anyone has unauthorized access during out of working-hours duration then the sensor will detect motion .If there is motion detected , this would stimulate the camera to record a video to the place in which there is unauthorized access. Accordingly, the alarm is sent to the guard, the image and livestream is displayed in the Google glass to the guard.

The iGuard system can speed up back up required to stop the violence in case of unauthorized accessed facilities. iGuard also can reduce the number of security guards required to scout in the facility. A proof of concept prototype is produced during the work on this project proofed that it is easy to be considered for commercialization.

# الملخص

إن تأمين وحماية مرافق العمل، المؤسسات أو الأعمال أو السياحة أو الجيش أمر مطلوب للغاية. تعتمد أنظمة إدارة الأمان الحالية على تعيين فرق من حراس الأمن مدعومين بمجموعة من كاميرات المراقبة وأجهزة المراقبة. تتحقق هذه الأنظمة من النقاط التي قام بزيارتها حراس الأمن في المبنى / المنشأة
بالكشف عنها والإبلاغ عن فشل او انحراف خطط التحقق الأمني. وفي الوقت نفسه، تتطلب هذه الأنظمة على الأقل حارس واحد لمراقبة شاشات الكاميرات في غرفة التحكم. بالإضافة إلى ذلك، فإن أنظمة حراسة الأمن المتنقلة والمتاحة حاليا؛ على سبيل المثال Pancomp، يراقب أداء حراس الأمن وليس تسهيل.
الحل المقترح لإدارة الأمن المحمول iGuard، يوفر نظام محمول قوي وفعال من حيث التكلفة لضمان تشغيل جميع العمليات الأمنية مع أقل جهد وزيادة الوعي. ينفذ iGuard المقترح نقطة إلى نقطة(P2P) أو نقطة الى عدة نقاط من الصور ومقاطع الفيديو من الكاميرا إلى نظارات قوقل الذي يرتديها حارس (حراس) الأمن. هذا مهم جدًا للتغلب على مشكلة بطارية نظارات قوقل الزجاجية التي قد تكون من أجل أن يتلقى حارس الأمان التنبيه والبث المباشر

يدمج iGuard تقنيات مختلفة تشمل الهاتف المحمول وكاميرا المراقبة ومستشعر PIR ونظارات قوقل
يتم إرسال البيانات الحسية وسجلات الكاميرا إلى شاشات التحكم الرئيسية في غرفة التحكم. يجب أن يحتوي النظام على جدول البحث الذي يحتوي على خارج ساعات العمل الطبيعية. وبالتالي، تشير أية بيانات حسية غير طبيعية إلى التحقق الفوري من سجلات الكاميرا عبر نظارات قوقل. يتطلب تطوير نظام iGuard توافر البنية التحتية للكشف عن الحركة، بالإضافة إلى كاميرا في مكان العمل أو المكاتب. سيتم تشغيل هذه الأجهزة للإحساس خارج ساعات العمل. سيتم الكشف عن البيانات العادية خلال ساعات العمل. ومن ثم، إذا كان أي شخص لديه وصول غير مصرح به خلال خارج ساعات العمل، فيقوم المستشعر بالكشف عن الحركة.
إذا كانت هناك حركة مكتشفة، فهذا من شأنه أن يحفز الكاميرا على اخذ صوره وعرض بث مباشر للمكان الذي يوجد فيه وصول غير مصرح به. وفقًا لذلك، يتم إرسال المنبه إلى الحارس، ويتم عرض الصورة والبث المباشر في نظارات قوقل إلى الحارس.

يمكن لنظام iGuard تسريع النسخ الاحتياطي المطلوب لوقف العنف في حالة وجود تسهيلات الوصول غير المصرح به . iGuardأيضا يمكن أن تقلل من عدد من حراس الأمن المطلوبة للاستكشاف في المنشأة.
يتم إنتاج دليل على النموذج الأولي للمشروع أثناء العمل على هذا المشروع، مما يؤكد أنه من السهل النظر إليه لأغراض تجارية.

# CHAPTER 1: INTRODUCTION

# Introduction

Securing work, institutional, business, tourism or military facilities is highly demanded. Current security management systems rely on hiring teams of security guards supported by a set of surveillance cameras and monitoring devices. These systems verify the visited points by security guards in the building/facility and report the deviation of security check plans. Meanwhile, these systems require at least one guard should monitor the cameras displays in a control room. In addition, the currently available mobile security guard systems; for example Pancomp, monitors the security guards performance not the facilities.

# Problem Statement

Lack of experience and shortage of security guards in addition to the violence and criminals experience in deceiving the surveillance cameras are increasing. The aim is to have one or few guards who can do many tasks to control and keep the building safe from any damage. The guard does not have to stay in one place to monitor. Therefore, we have to add mobility to guards we need to use some tools such as Google Glass and Motion sensors as a mobile security Guard.

# Project Goals and Objectives

The project goal is to add mobility to guards, so one guard can act like a team of guards and does not have to stay in a room to monitor. Technical goal is to produce prototype to embed simple infrastructure equipped with PIR sensors and cameras readings and make them compatible across multiple platforms along with wearable devices such as Google glass.
In summary, the proposed iGuard system is an attempt to proof the concept of using multiple technologies to produce working prototype of cost effective iGuards system. This system reduces human resources required for security operations as few security guards can monitor the situation from anywhere whilst at the same time increases their monitoring efficiency

## The Project Objectives are:

1. To allow guards to perform regular check of multiple locations in the building whilst they are away from the monitor.
2. To create innovative alarming system in order to notify the guard both images and Livestream for any changes occur to the normal status in case of unauthorized access.
3. To use explore commerciality of display tools for the information retrieved to the security guard; for example, using Google glass.

# Proposed Solution

iGuard, integrates multiple technologies. iGuard includes motion detector board and a main controller using Arduino technology. The PIR sensor board detect the motion for example in the offices, labs and servers room also in order to monitoring room then sends all the information to the main controller which sends the data to a web server and its database. Hence, if there is changes in the normal status by unauthorized access especially out of working hours, the security guard will receive alarm indicate that from the second part of the network, which displays the information retrieved to the security guard. One of the most functionalities that the proposed system offers is to let the guards keep their mobility and give them the ability to move and monitor at the same time. What we aim for in mobile security Guard is to produce a prototype that integrates Arduino technology and Google glass with mobile application to enhance security management applications.


*Figure 1 iGuard: Mobile Security Guard System*

## Project Scope

The proposed solution requires infrastructure to embed the Motion sensor board and a main controller using Arduino technology. In addition, the proposed solution also requires a surveillance camera to be fixed in the facility. For this, the work in this project included preparing a physical prototype of an office to fix the sensors and camera and hence connect them to the system. Accordingly, the system shall detect if there is a motions in the location, then alarm is sent to the guard with image and Livestream from the location where the sensor detected abnormal data and displayed in the Google glass to the guard. On the other hand, the proposed system does not control door access security system.

# Hardware and Software Tools

**Hardware:**

1. Computers
2. Mobile phones
3. Google glass
4. Camera
5. Arduino sensor and toolkit.

**Software:**

1. Android studio
2. Camera app
3. Arduino IDE
4. Android studio built in testing tool.
5. MyGlass app

# Cost

Google glass = 3000 $
Arduino = 112.34 $

# Timeline

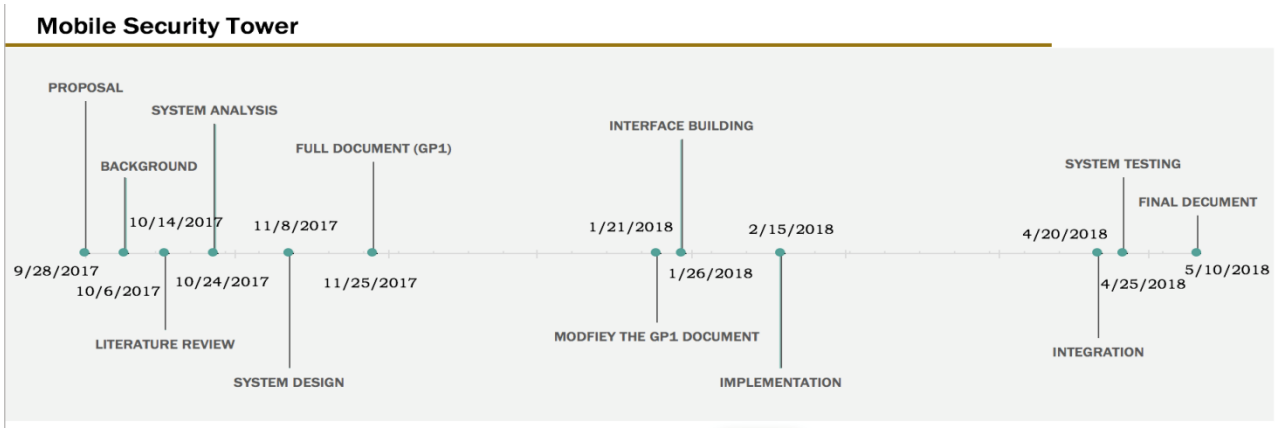



*Figure 2: TimeLine graph*

*Table 1:TimeLine table*

| DATE | TASK | DURATION | FINISH |
|---|---|---|---|
| 28-Sep | Proposal | 8 days | 5-Oct |
| 6-Oct | Background | 8 days | 13-Oct |
| 14-Oct | Literature review | 10 days | 23-Oct |
| 24-Oct | System Analysis | 15 days | 7-Nov |
| 8-Nov | System Design | 17 days | 24-Nov |
| 25-Nov | Full Document (GP1) | 17 days | 11-Dec |
| 21-Jan | Modfiey the GP1 Document | 5 days | 25-Jan |
| 26-Jan | Interface Building | 20 days | 14-Feb |
| 15-Feb | Implementation | 63 days | 19-Apr |
| 20-Apr | Integration | 5 days | 24-Apr |
| 25-Apr | System Testing | 15 days | 9-May |
| 10-May | Final Decument | 4 days | 13-May |

# Roles and Responsibilities

*Table 2:Roles and Responsibilities*

| Role | Responsibilities | Participant(s) |
|---|---|---|
| Project Leader | • Manage meetings<br>• Tasks distribution in each phase<br>• Monitor Progress and make sure that all tasks submitted on time<br>• Collect team members' work and combines all the parts of the project<br>• Notify the supervisor about any team problems<br>• Ensure that updates are made to the report after reviewed by the supervisor | Tahani Alghanem |

| | | |
|---|---|---|
| | • Submit deliverables | |
| Analysis | • Collect requirements<br>• Structures the requirements into SRS<br>• Develop required system models | All group members |
| Design | • Implements the Pseudo code<br>• Write the programming code<br>• Perform unit testing | All group members |
| Implementation | • Develop the design that the Analyst team produced<br>• Data design (Database schema design)<br>• Writing pseudo code.<br>• Software interface design | All group members |
| Testing | • Perform unit testing, system testing<br>• Ensure the project meets the specifications in each phase<br>• Produce a report showing what part should be modified<br>• Try to repair any problem appears | All group members |
| Domain Experts | • Meet with our supervisor Dr.Ghada weekly to review each phase | All group members |

# CHAPTER 2: LITERATURE REVIEW

# Cameras in Surveillance Connaincense

Cameras are used for monitoring everywhere on street corners, schools, malls and bank ...etc. Video cameras are briefly reviewed in this section to explore the history and evolution of video surveillance cameras.

Thomas Edison and William Dickson produce the first film cameras in 1880. They were two inventors who worked together they were close to the movie camera problem from two different ends. Edison worked on his Kineto phone where Dickson focused on his own version, which is Kineto graph. They work together and they invent the first public presentation of a motion picture in 1893. Within a few years, commercial motion pictures were being produced and made and exhibited across America, and the seed for video surveillance had been planted.

*Figure 3: Thomas Edison and William Dickson*

1939: First time that the diminutive portable cameras appear in advanced warfare such as the Univex 8mm.they could be carried comfortably in one hand, and were operated through vault rolling. Hidden surveillance was made possible. because in that time in history, a camera could be used without any attention

*Figure 4: Portable cameras*

In 1942: Closed Circuit Television (CCTV) is first used in Germany. German scientists developed the technology so that they could monitor the launch of V2 rockets. Afterward, this kind of video surveillance was used in the United States during the testing of Atomic Bombs.

In 1951, the videos tape recorder (VTR) was invented to record live images from a television camera it used magnetic recording strip. After five years, this technology would become commercially available, and it used CCTV to record surveillance for later showing.

*Figure 5: Video Tape Recorder*

1960: Temporary cameras was used to monitor Thai royalty in England. it was set up in Trafalgar Square in London to help protect visiting royalty from crowds.

*Figure 6: Temporary cameras*

In 1965: After a couple of years' Public surveillance cameras became more usual. Press reports from the time indicate that police had adopted the role of cameras in many public places.

1969: The first video home security system is born. Marie Van Brittan Brown received a patent on her system which consisted of four peepholes and camera that could be moved to look through any one of them. The camera would transmit its images to a proctor.

1970s: CCTV usually they use it in the non-government market. Bank and retailers started to use CCTV to add security measure to their property. Till these days, we are using it but in advanced way.

*Figure 8:Nanny Cam*

1976: Charge-coupled device (CCD) technology it is a camera that can be used in low light situations.it works round-the-clock as possible. 1990s: ATMs have cameras installed to record all transactions. [1]. In1992, The first camera to monitor a baby from home in any other place was "Nanny Cam". It was camera technology to allow surveillance for smaller high-resolution, parents start using it to keep an eye on their children covertly.

*Figure 9:Public surveillance cameras*

1993: the first attack on the World Trade Center all profile location the number of monitoring has been increased. people be more aware of the possibility of terrorist attacks. The use constant monitoring in sporting events and other different places.

1996: After CCTV, they invented IP camera. The IP camera could send and receive information between multiple computer networks. This drive to later webcam and stop using CCTV.

*Figure 10:Freed camera*

Webcam monitoring become more common in 2011, the second attack on the World Trade Center 2011, tends the public to have a more personal safety. Nowadays face recognition and other digital advances became a more necessary thing.

Today: Utilizing the internet and wireless communication, video surveillance can now be used and viewed from anywhere in the world. This is used to great effect in personal residences, as homeowners are now capable to utilize inexpensive video surveillance as a form of home security.

## Immune Chromatographic Diagnostic Test Analysis Using Google Glass

A biomedical sensing application uses a wearable computer (Google Glass). it uses this technology because it is the fastest for(RDTs) tests Google glass improves and make it fast for individual tracking of dangerous conditions or diseases. It used for monitoring the public health and fastest response . First the user must take a picture for RDT through google glass camera and using QR code .This application can detect and find the RDT type .Finally it retrieve the result to google glass.

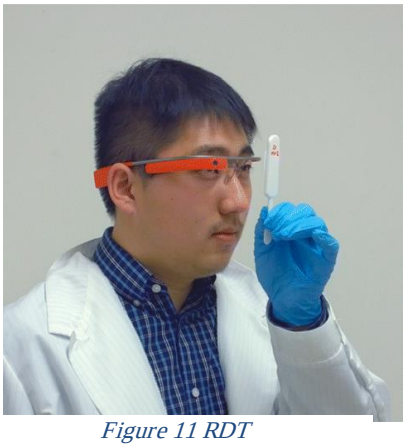


*Figure 11 RDT*

## Texting While Driving Using Google Glass

Texting while driving is common these days but it is risky so they use google glass which is a wearable device it displays the text on the glasses . This appliction can improves the driving performence and reduse the risk of using the phone while driving .


*Figure 12:Texting while driving using Google Glass*

## Google Glass In Pediatric Surgery: An Exploratory Study

This appliction use google glasses to help the educating of systems without a lot of costly gear (eg, video headsets, transmission wiring and screens). the makers presumed that Google Glass would be the right path for students to be the effective tool for self-audit though the more lumbering GoPro would be utilized to store up a video library of basic operations. This app can use Google Glass remotely to  educate trainees and  also it can help other trained surgeons that may be running into a problem and need extra help they can take a picture while working on surgery so that can reduce the medical faults.


*Figure 13:Google Glass in pediatric surgery*

## Xbee Wireless Sensor Networks For Temperature Monitoring

XBee Wireless Sensor Networks is a wireless sensor network(WSN) it is a system where it senses the temperature in a building. the use of the network is to manage the air condition systems .the goal of this system is to save the energy cost. it has a Guard interface to show the current and previous temperature readings in different rooms. each (WSN) has a microcontroller on Arduino board .also it can be accessed by a web browser.

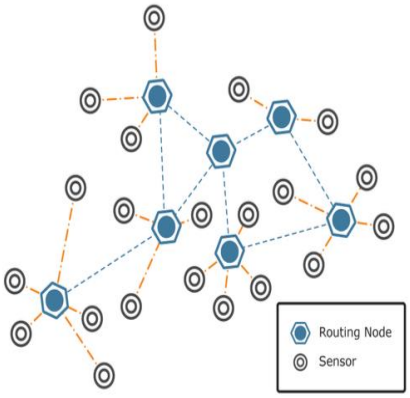



*Figure 14:Ad hoc wireless sensor Network*

## Iot Status Monitoring With Ar On Google Glass

A system that uses the (AR) capabilities of google glass connect them with the idea of internet of things .the system gather the information form an exact environment and store it the Guard can view the stored information by the web or they can use a QR codes scanning on a custom app and then the google glass interface view a real-time infographic for the information that are related to the scanned QR code it is reflected by the web interface .

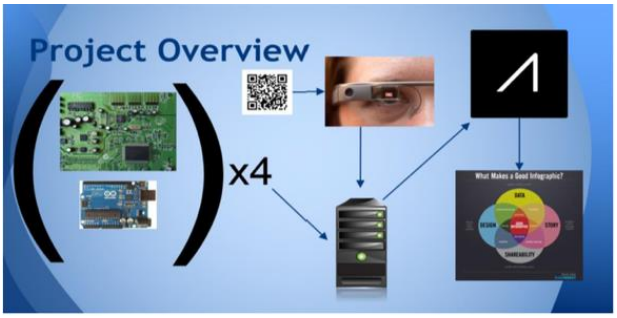



*Figure 15:Internet of Things Status Monitoring*

# ARDUINO TECHNOLOGY

“In 2005 the Arduino born in Italy by the student Massimo Banzi as a project, the Arduino is the concepts of free hardware and software”. The Arduino was designed to everyone interested in creating interactive objects everyone can use it such as artists, designers, hackers and physicists by connecting the object with it.


*Figure 16:Microcontroller board*

The first Arduino was a microcontroller board based on the ATtiny85 it supports the microcontroller, it can be connected to the computer using USB cable, power bank or battery. it called Arduino Gemma. After that, the Arduino 3D printer came out. The first official Arduino 3D printer is Arduino Materia 101. It uses a bulker specifically Arduino Mega 2560 for controlling 3D printers.

*Figure 17:Arduino Materia*

## 2.2 COMPETITIVE ANALYSIS

*Table 3: Competitive analysis*

| Applications<br><br>Features | iGuard | Texting while driving using google glass | XBee Wireless Sensor Networks for Temperature Monitoring | IOT Status Monitoring with Augmented Reality on Google Glass |
|---|---|---|---|---|
| **Available for android** | √ | √ | | √ |
| **Require Guard training** | √ | | | |
| **Ease of use** | | √ | √ | √ |
| **Detect motion** | √ | | | |
| **Sense Temperature** | | | √ | √ |
| **Sense sound waves** | | | | √ |
| **Sense light intensity** | | | | √ |
| **Alert Guard if needed** | √ | √ | | |
| **Display information on Google Glass** | √ | √ | | √ |

As it appears in the table above, we noticed that most systems are easy to use. Also, the only system that didn't support displaying information on the Google glass was XBee wireless sensor network. The common features in all systems were availability for android, easy to use and display information on Google glass.

# CHAPTER 3: SYSTEM ANALYSIS

# Software Requirements and Specifications

## Functional requirements:

1. The system shall allow guard to Register.
2. The system shall allow guard to login/out.
3. The able adminstartor shall be to manage accounts.
4. The system shall allow guard to add location.
5. The system shall allow guard to select one / multiple location to monitor.
6. The system shall detect motion.
7. The system shall take a picture if the sensors detect motion.
8. The system shall alert the guard by display the image to the security guard in the Google glass if the sensors detect motion.
9. The system shall alert the guard by display a Livestream to the security guard in the Google glass if the sensors detect motion.

## Non-functional requirements:

1. The system shall perform in a proper time constraint.
2. All the features of the system shall operate properly.
3. The system shall fast response to the sensory data.
4. The system shall be available 24/7.
5. The system shall allow guards to access the information available to them only.
6. The language of the system interface shall be in English.
7. The system shall be usable to the security guard.
8. The system shall be Compatible with the google glass and android mobile phone.
9. The system shall daily remove the faults effect to insure the quality of the system.

# Use Case Diagram

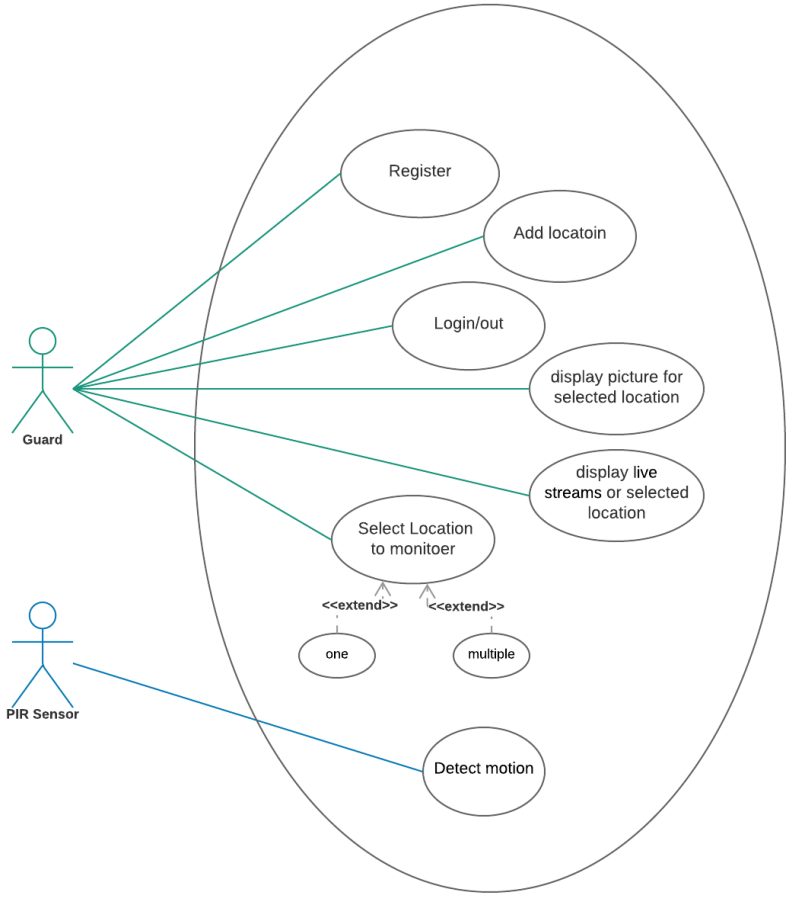



*Figure 18:Use case diagram*

## EXPANDED USE CASE

## 1. Register

| **Use case** | **Register** |
|---|---|
| **Actors** | Guard(Initiator) |
| **Purpose** | Register in to the application(system) |
| **Overview** | The Guard Register<br>his information. |
| **Type** | Primary |
| **Cross Reference** | R1 |

**Typical Course of Events**

| **Actor Action** | **System Response** |
|---|---|
| 1. this use case begin when a guard want to register in to the system . | |
| | 2. The system displays the registration form. |
| 3. the guard fill the registration information. | |
| | 4. The system store the registration information. |

*Table 4:Use case description for Register*

## 2. Add location

| Use case | Add location |
|---|---|
| **Actors** | Guard (Initiator) |
| **Purpose** | Add a new location to monitor . |
| **Overview** | The Guard add a new one or multiple locations to monitor. |
| **Type** | Primary |
| **Cross Reference** | R4 |

Typical Course of Events

| Actor Action | System Response |
|---|---|
| 1. this use case begins when a Guard want to add a new location to monitor. | |
| | 2. The system displays the location form. |
| 3. the Guard fill the location information (city, building name, room number, camera IP, sensors ID). | |
| | 4. The system store the location information. |

*Table 5:Use case description for Add location*

## 3.Select location to monitor

| Use case | Select location |
|---|---|
| **Actors** | Guard |
| **purpose** | Select the location he wants to monitor |
| **Overview** | The guard select one/multiple location/s he wants to monitor |
| **Type** | primary |
| **Cross Reference** | R5 |

**Typical Course of Events**

| Actor Action | System Response |
|---|---|
| 1. the guard select the location. | |
| | 2. The system displays the location monitoring options. |
| **Alternatives:**<br>**Line1: select one or multiple locations** | |

*Table 6:Use case description for Select location*

## 4. Detect Motion

| **Use case** | Detect **Motion** |
|---|---|
| **Actors** | Sensor |
| **purpose** | Read PIR Motion detection sensors |
| **Overview** | The guard select the location he wants to monitor then the sensor detect in there is any motion. |
| **Type** | primary |
| **Cross Reference** | R6 |

**Typical Course of Events**

| Actor Action | System Response |
|---|---|
| 1. the guard select the location. | |
| | 2. The system displays the location monitoring options. |
| **3.** The guard select start monitoring. | |
| 4. the sensor in the location detect if there is a motion. | |
| | 5. The system listen to the sensor data. |

*Table 7:User case description for detect motion*

## 5. Display Picture

| Use case | Display Picture |
|---|---|
| **Actors** | Guard |
| **purpose** | Display picture of selected location |
| **overview** | The guard select to view a picture that was taken in a location. |
| **Type** | Primary |
| **Cross Reference** | R8 |

**Typical Course of Events**

| Actor Action | System Response |
|---|---|
| | 1. The system alert the guard |
| 2. The guard select to display the picture | |
| | 3. The system display the picture |

*Table 8: Use case description for Display picture*

## 6. Display Livestream

| Use case | Display live stream |
|---|---|
| **Actors** | Guard |
| **purpose** | Display live stream of selected location |
| **overview** | The guard select to view a live stream for this location. |
| **Type** | Primary |
| **Cross Reference** | R9 |

**Typical Course of Events**

| Actor Action | System Response |
|---|---|
| | 1. The system alert the guard |
| 2. The guard select to display the live stream | |
| | 3. The system display the live stream |

*Table 9:Use case description for Display live stream*

# Conceptual Model

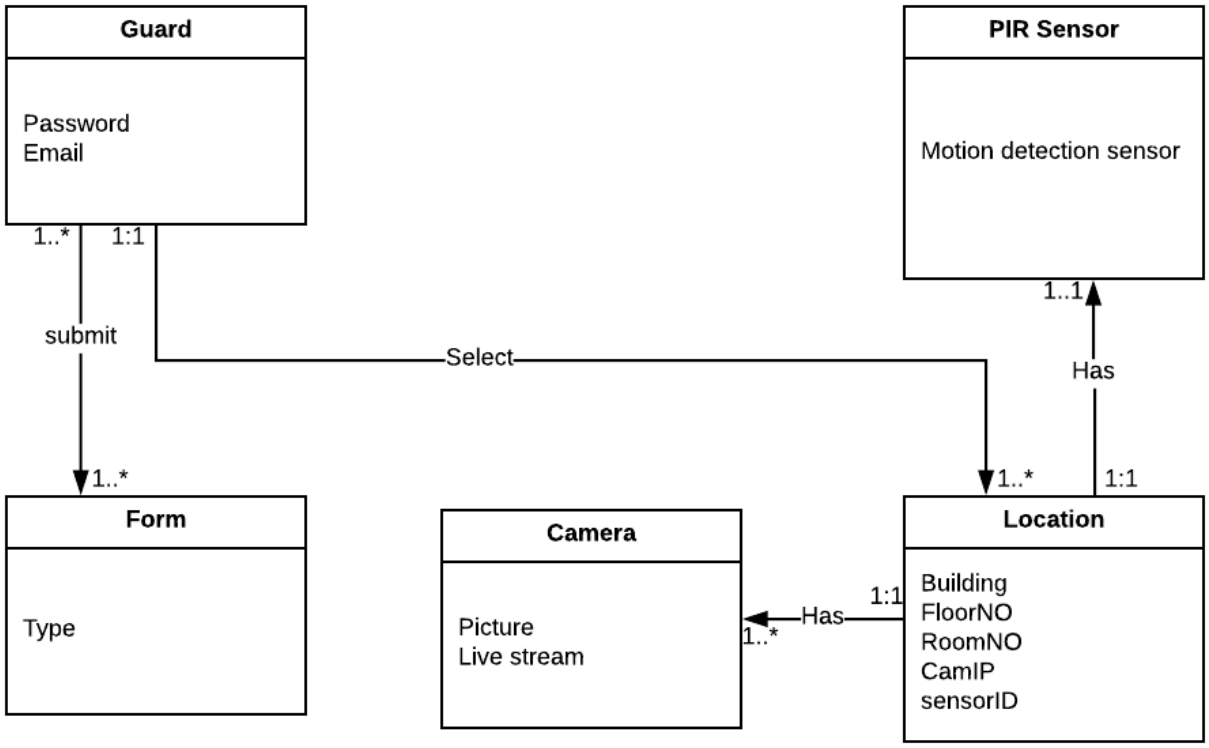



*Figure 19:Conceptual Model*

# Sequential Diagrams

## Register sequence

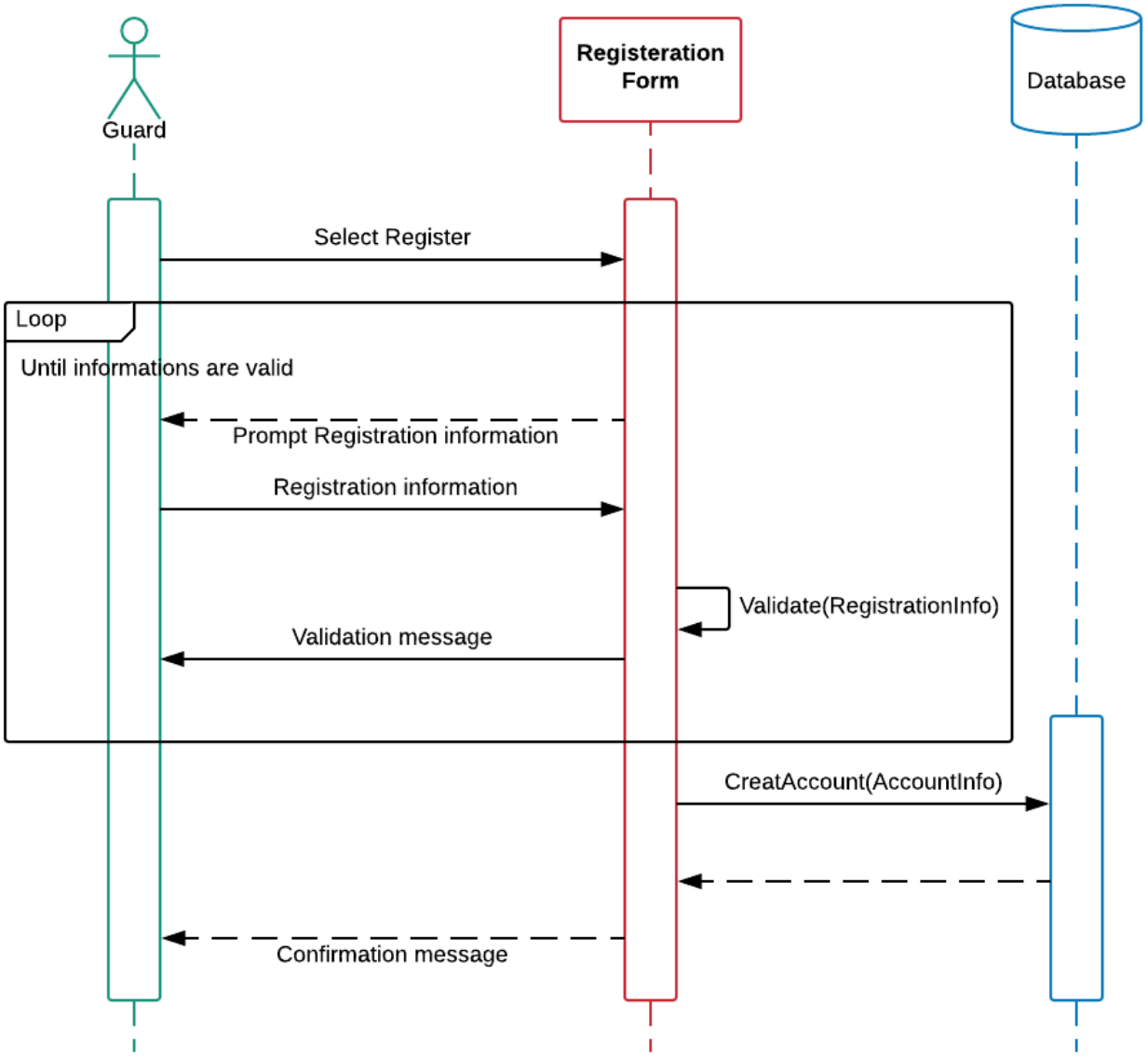



*Figure 20:Register sequence*

## Add location sequence

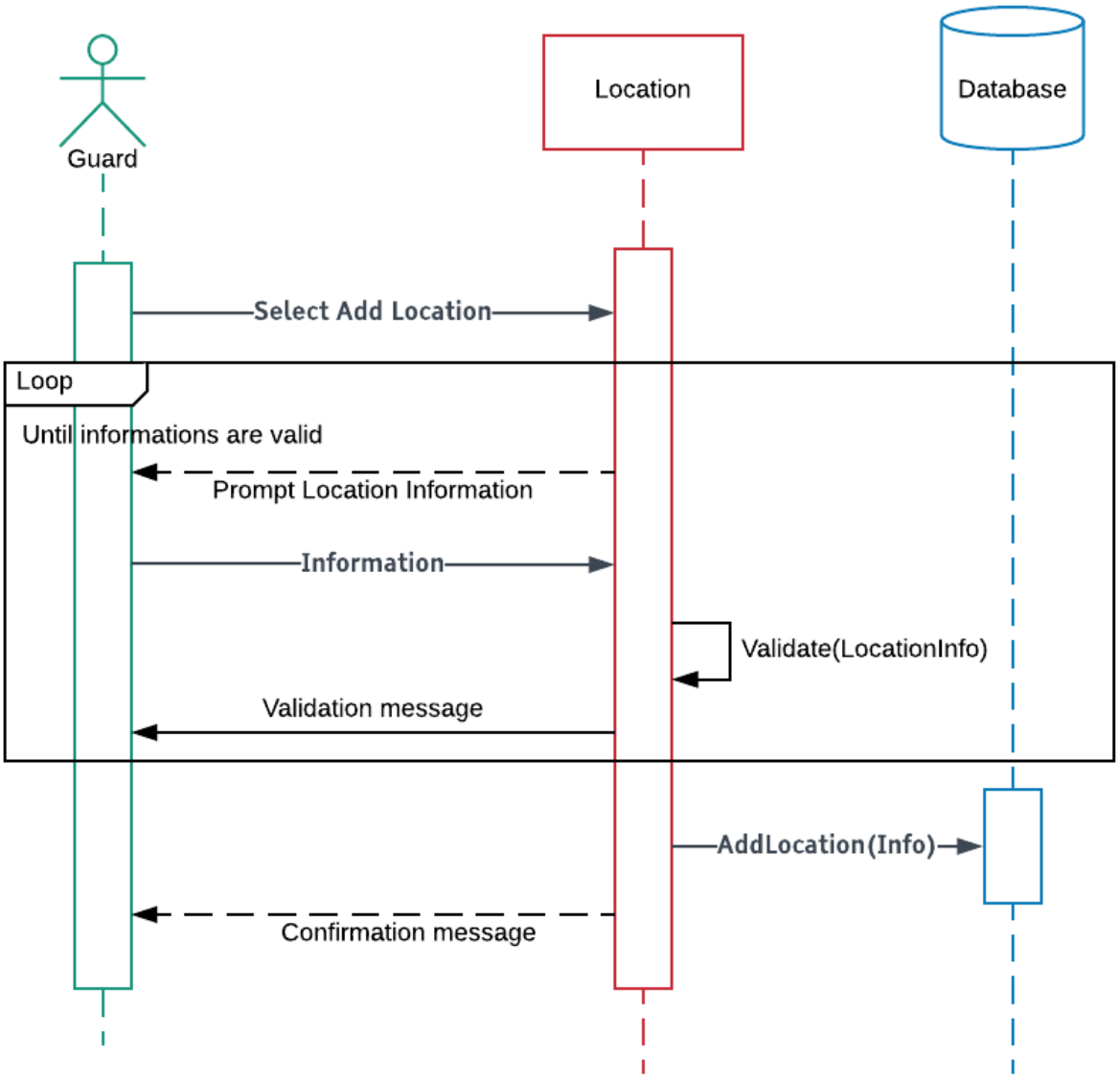



*Figure 21:Add location sequence*

## Detect motion sequence

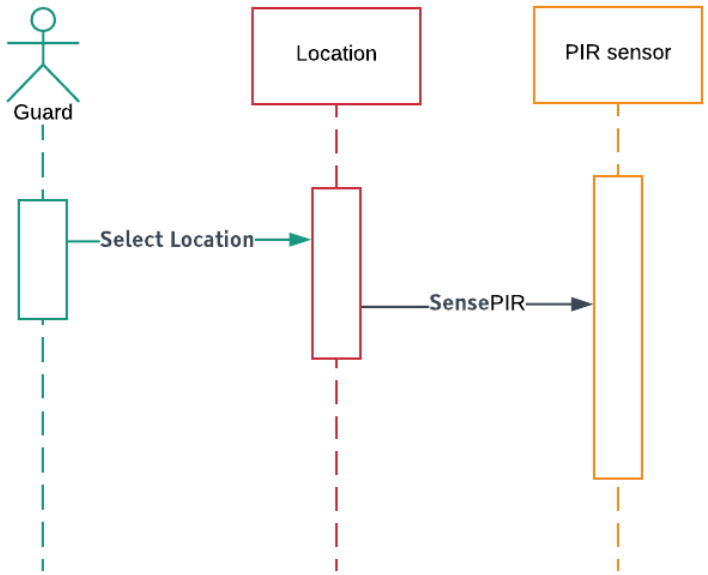



*Figure 22:Detect motion sequence*

## Display Picture sequence

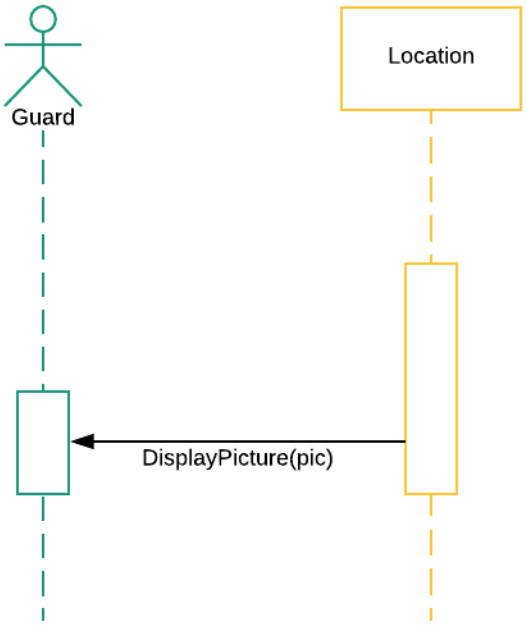



*Figure 23:Display Picture sequence*

## Display Live Stream sequence

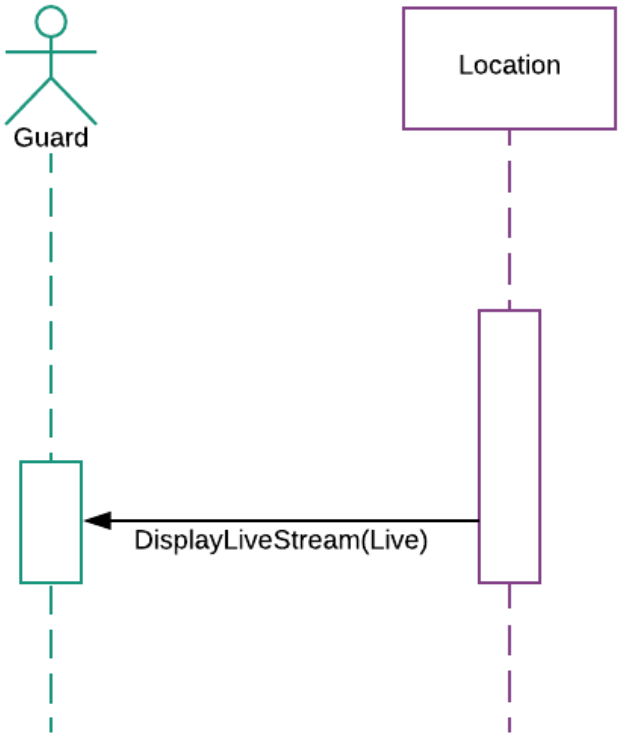



*Figure 24:Display Live Stream sequence*

# CLASS DIAGRAM

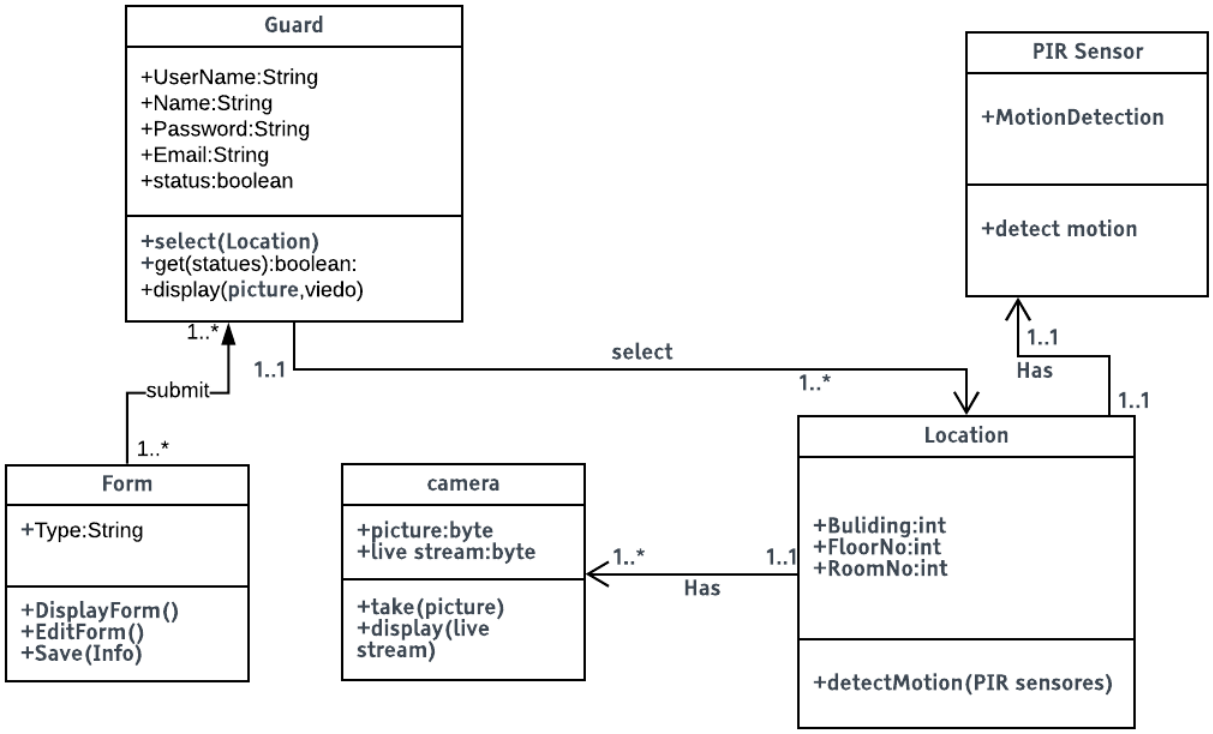



*Figure 25:class diagram*

# CHAPTER 4: SYSTEM DESIGN

# Architectural Design:

In Mobile security Guard system, we focus on providing the maximum availability of the services that the system provides: reliability, fast response and execute, available, usability, and efficiency.
after analyzing the system architecture, we think so using A layered pattern will best serve the system to meet up with its services.

The layered architecture for Mobile security Guard system is composed of the following layers:

1. Presentation Layer
   Presentation of the web pages, UI forms and end Guard interacting API's

2. Business Layer
   The logic behind the accessibility, security and authentication happens in this layer, middle ware and other various request interceptors to perform validations.

3. Persistent layer
   This is the presentation layer for the Data. This includes the DAO (Data Access Object) presentation, ORM (Object Relational Mappings) and Other modes of presenting persistent data in the application level. In more meaningful words this demonstrates the persistent data in RAM. Which usually stays in Disks at the below layer.

4. Database Layer
   Simple Databases expanding up to SANs (Storage Area Networks).

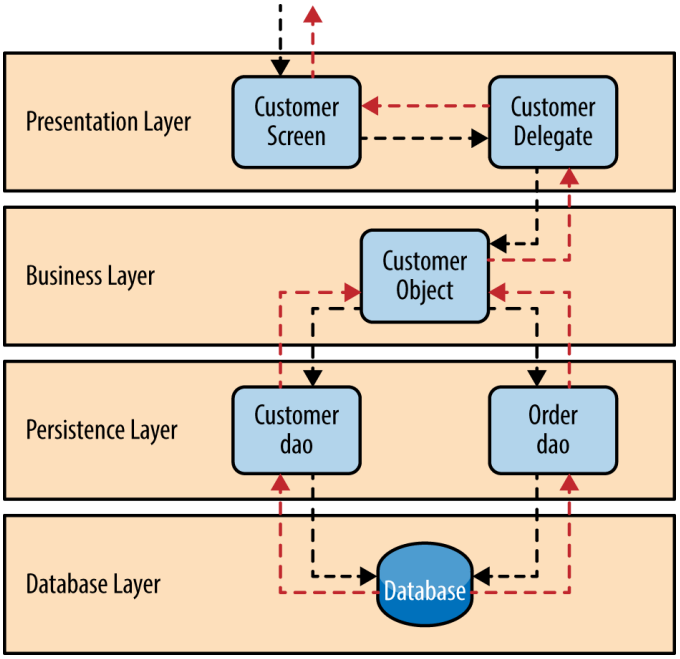



*Figure 26:layered architecture*

# Component Design

Component design is a detailed design of all the systems functionality. It is simplifying the system's complexity by using pseudo code. Pseudo code shows the logic of the tasks without specifying the syntax of the language (programing language independent).

# Detailed System Design

## Detect motion

**Classification**: function
**Definition**: The system listens to the data sent from Arduino if the data tells there is a motion detected a picture will be taken from the camera and sent to google glass.
**Responsibility reference**: R6
**Composition**: none

### Pseudocode:

1.**Begin**
2.**IF** guard select location **THEN**
3.**IF** guard select start monitoring **THEN**
4.**GET** PIR sensor readings
5.**End IF**
6.**End IF**

## Add Location

**Classification:** function
**Definition**: The Guard added a new location one or multiply location to monitoring by the guard.
**Responsibility reference**: R4
**Composition**: none

### Pseudocode:

1.**Begin**
2.**IF** guard selects Add location **THEN**
3.**Display** location information form
4.**Read** location information
**4.Store** the information in the database
**5.Display** Confirmation message
**6.End IF**

## Select Location

**Classification**: function
**Definition**: The guard select one/multiple location/s he wants to monitor
**Responsibility reference**: R5
**Composition**: none

### Pseudocode:

1.**Begin**
2.**IF** guard selects location **THEN**
3.**Retrieve** location information from database
4.**Display** location information
5.**End IF**

## Display picture for selected location

**Classification**: function
**Definition**: A picture of selected location displayed on google glass.
**Responsibility reference**: R8
**Composition**: none

### Pseudocode:

1.**Begin**

2.**IF** guard select location **THEN**
3.**IF** guard select start monitoring **THEN**
4.**IF** motion detected **THEN**
5.**Take** a picture
6.**Send** to google glass.
7.**End IF**
8.**End IF**
9.**End IF**

## Display live stream for selected location

**Classification**: function
**Definition**: A live stream of selected location displayed on google glass.
**Responsibility reference**: R9
**Composition**: none

## Pseudocode:

1.**Begin**

2.**IF** guard select location **THEN**
3.**IF** guard select start monitoring **THEN**
4.**IF** motion detected **THEN**
5**. Display** a live stream
6.**Send** to google glass.
7**.End IF**
8**.End IF**
9**.End IF**

# Interfaces Design

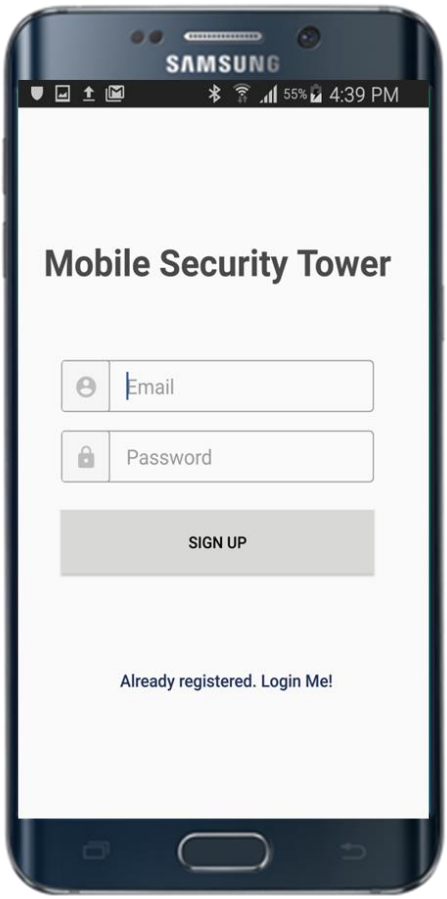



*Figure 27:Register interface*


*Figure 28:startup interface*

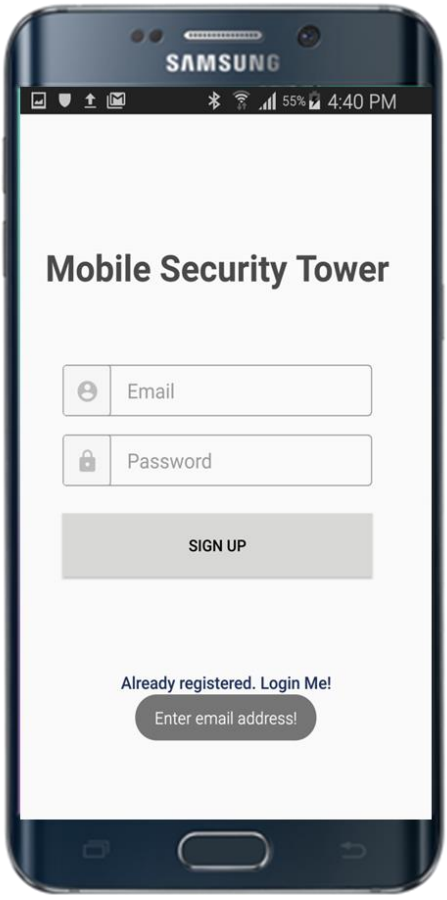



*Figure 30:error message*

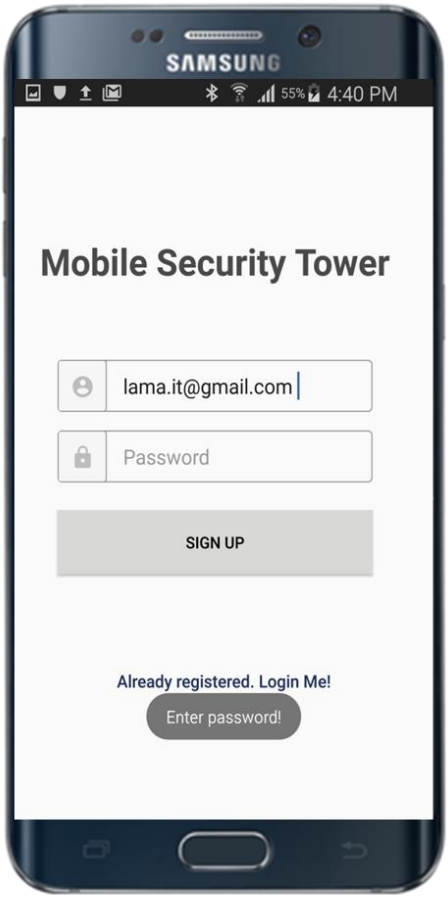



*Figure 29:error message*

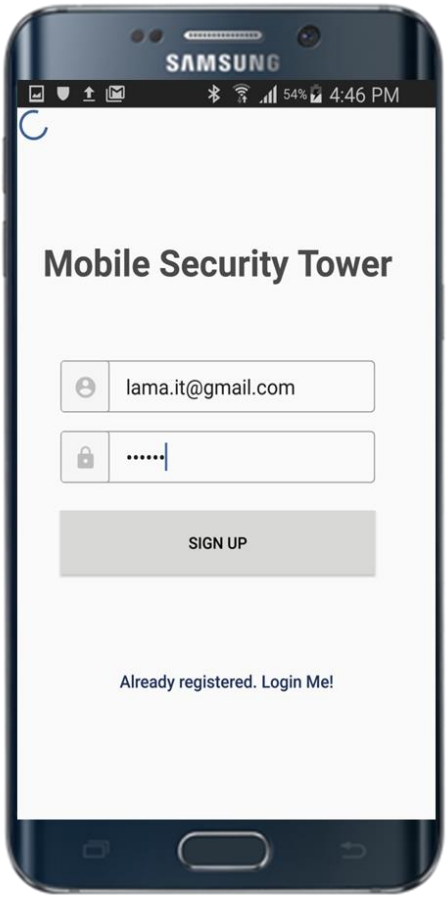



*Figure 32:progress bar*

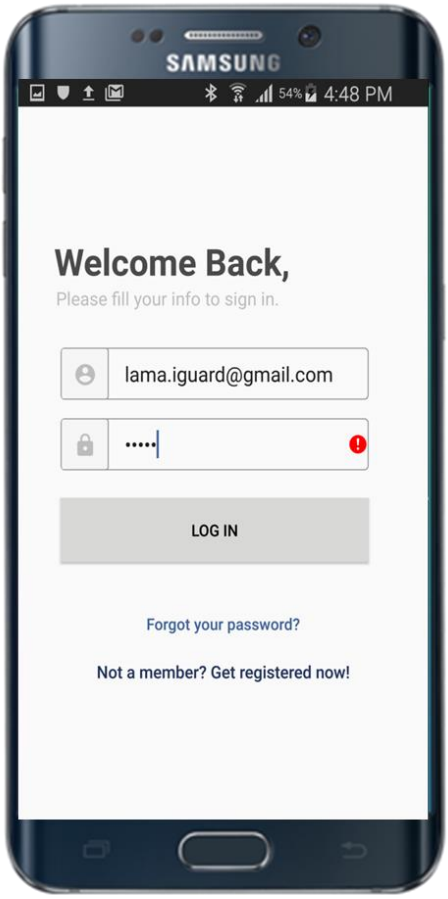



*Figure 31:login interface*

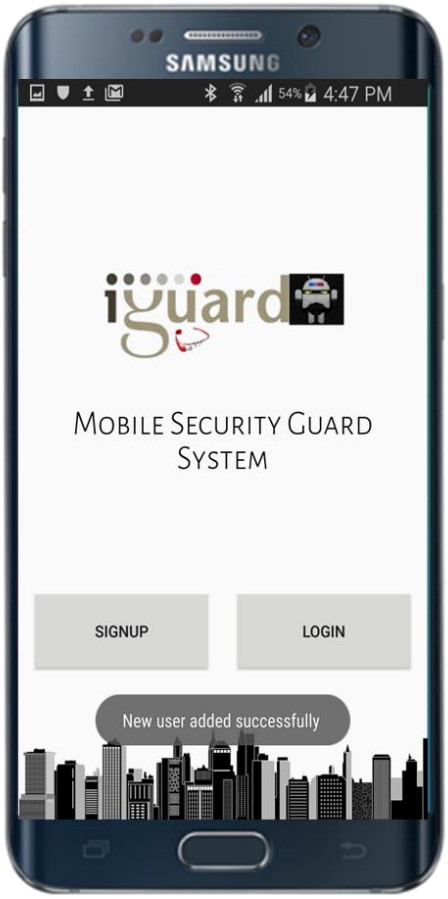



*Figure 34:New user confirmation message*

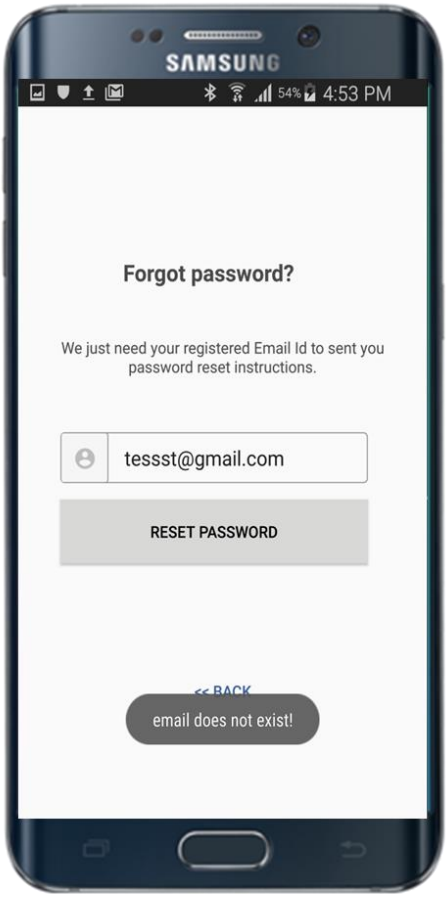



*Figure 33:Retrive password interface*

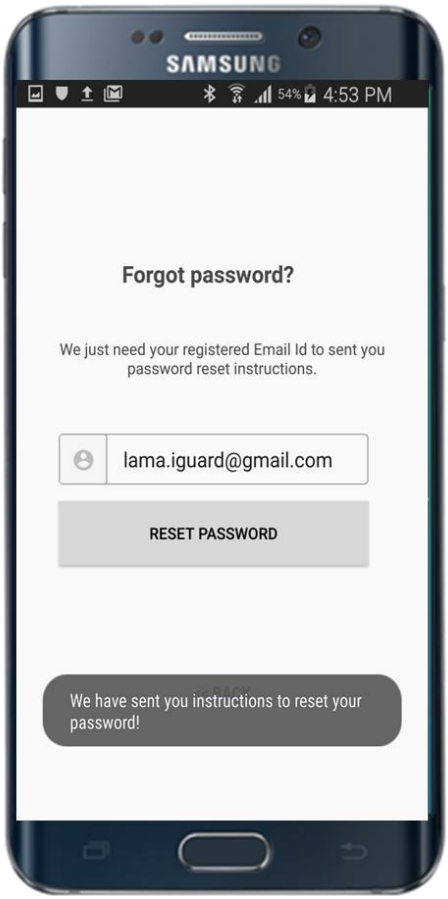



*Figure 36:Retrive password confirmation message*

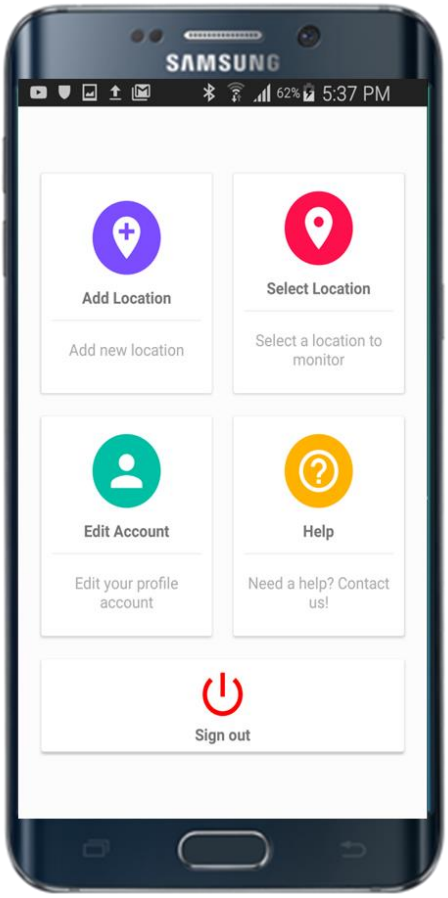



*Figure 35:Home page interface*

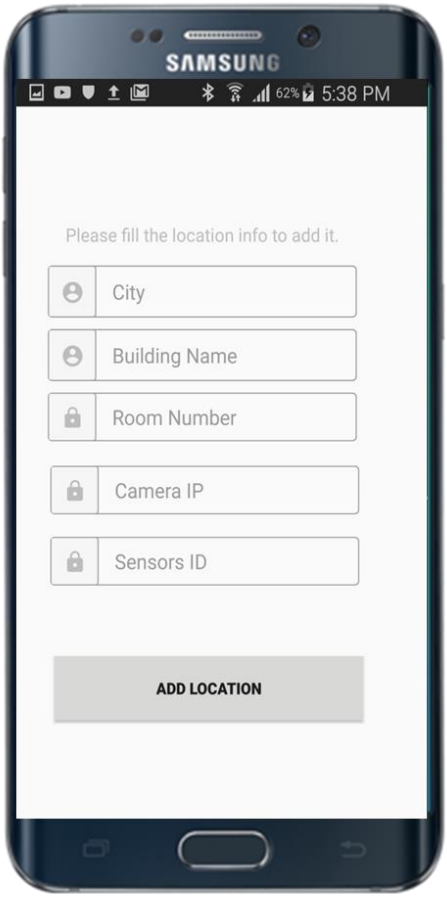



*Figure 38:Add location interface*

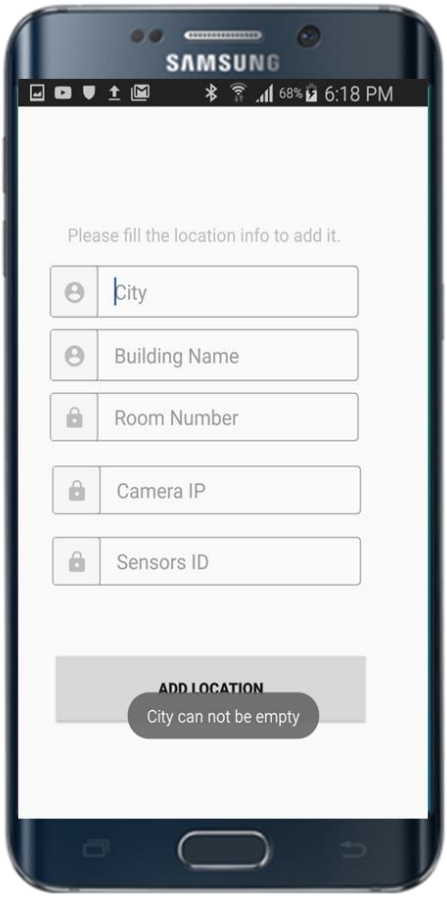



*Figure 37:error message*

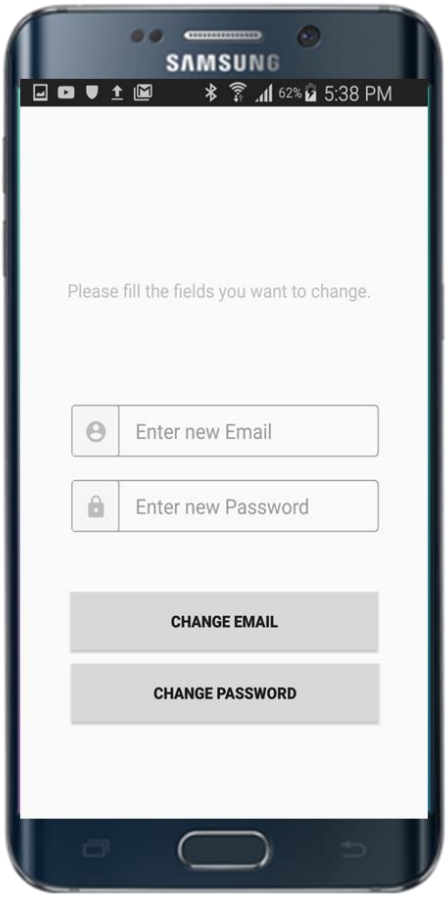



*Figure 39:Edit account interface*

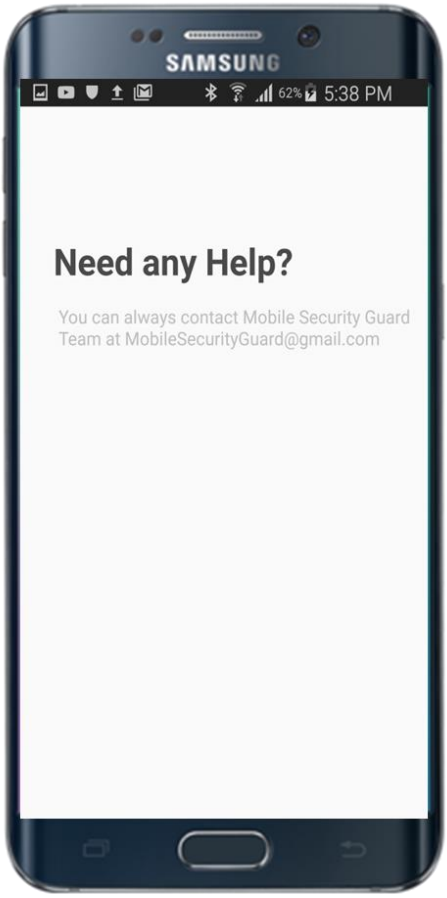



*Figure 40: Help interface*

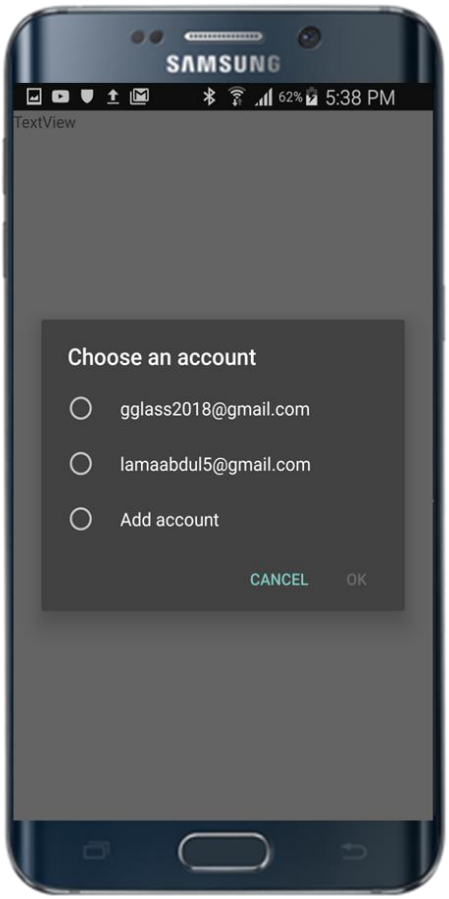



*Figure 42:Authentication interface*

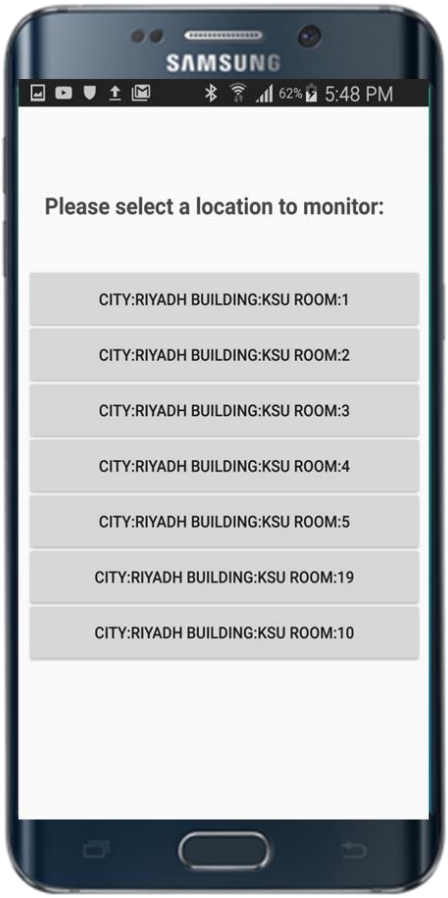



*Figure 41:Locations interface*


*Figure 44:Connecting to google glass confirmation*


*Figure 43:location options*

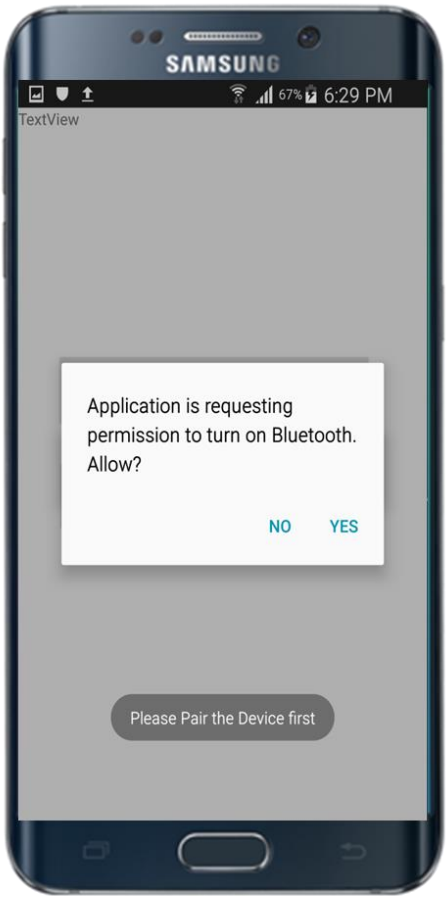



*Figure 45:Bluetooth access permission*

# Components Transactions Model


*Figure 46:Components Transactions Model*

# CHAPTER 5: IMPLEMENTATION AND SYSTEM INTEGRATION

## Software and Hardware:

*Table 10 Software used*

| Software | Usages |
|---|---|
| **Android studio** | Android studio is an IDE application that can create application using *java* and other languages for android devices. We use it to create the application to enable the guard to sign up and add location to monitor it. |
| **Arduino IDE** | This application is used to run and view the sensors reading using Arduino hardware and PIR sensor it uses *C* Language. |
| **Foscam application** | This application is used to setup the camera and to view the camera live stream. |
| **MyGlass application** | This application is used to setup the Google glass. |

*Table 11 Hardware used*

| Hardware | Usages |
|---|---|
| **Mobile phone** | This hardware used to run the application that let the Guard to register, select location etc.. |
| **Laptop** | This hardware used to run the code of each hardware, to let us use android studio and to integrate this hardwares together. |
| **Foscam camera** | This hardware used to take a snapshot if abnormal reading of the sensors detected. |
| **Google glass** | This hardware is wearable by the guard to view the picture and live stream that is taken by the camera. |
| **Arduino** | This hardware is used to detect motion. |

# Code Portions of The Contribution

- **Arduino**

Fist we connect PIR motion detection sensor with Arduino UNO as shown in figure [47] to use it for motion detection; then we wrote a code to read the sensor data, if there is a motion the data will be sent to the application.

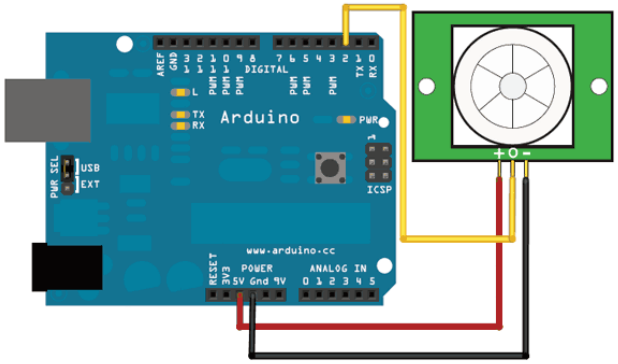



*Figure 47:PIR sensor connection*

```
////////////////////////////
//SETUP
void setup(){
  Serial.begin(9600);
  pinMode(pirPin, INPUT);
  digitalWrite(pirPin, LOW);

  //give the sensor some time to calibrate
    for(int i = 0; i < calibrationTime; i++){
      delay(1000);
      }

    delay(50);
  }

////////////////////////////
```

```
void loop(){

     if(digitalRead(pirPin) == HIGH){
       if(lockLow){
         //makes sure we wait for a transition to LOW before any further output is made:
         lockLow = false;
         Serial.println("motion detected");
         delay(0);
         }
         takeLowTime = true;
       }

     if(digitalRead(pirPin) == LOW){

       if(takeLowTime){
        lowIn = millis();          //save the time of the transition from high to LOW
        takeLowTime = false;       //make sure this is only done at the start of a LOW phase
        }
       //if the sensor is low for more than the given pause,
       //we assume that no more motion is going to happen
       if(!lockLow && millis() - lowIn > pause){
           //makes sure this block of code is only executed again after
           //a new motion sequence has been detected
           lockLow = true;
           delay(0);
           }
       }
  }
```

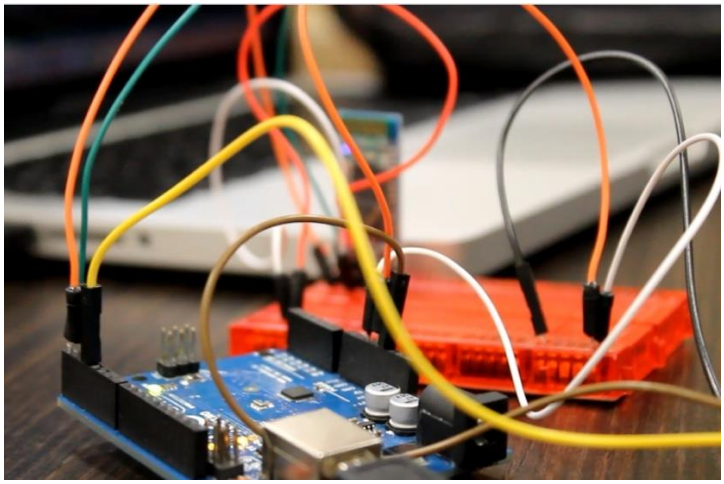


*Figure 48:Ardunio board*

*Figure 49:PIR sensor*

- **Listen for data sent from Arduino**

  In order to Listen for the PIR sensor data we connected the system with the Arduino then make a thread to listen for the data that will be sent from the Arduino.

```
void beginListenForData()
{
   final Handler handler = new Handler();
   final byte delimiter= 10;

   stopThread = false;
   readBufferPosition=0;
   buffer = new byte[1024];
   thread  = new Thread(new Runnable()
   {
      public void run() {
         while (!Thread.currentThread().isInterrupted() && !stopThread) {
            try {

               int byteCount = inputStream.available();
               if (byteCount > 0) {
                  byte[] rawBytes = new byte[byteCount];
                  inputStream.read(rawBytes);
                  for (int i = 0; i < byteCount; i++) {

                     byte b = rawBytes[i];
                     if (b == delimiter) {

                        byte[] encodedBytes = new byte[readBufferPosition];
                        System.arraycopy(buffer, 0, encodedBytes, 0, encodedBytes.length);

                        value = new String(encodedBytes, "US-ASCII");
                        readBufferPosition = 0;

                        handler.post(new Runnable() {
                           public void run() {
                              message(value);

                           }
                        });

                     } else
                        buffer[readBufferPosition++] = b;
                  }
               }
            } catch (IOException ex) {
               stopThread = true;
            }
         }
      }
   });

   thread.start();

}
```

- **Mirror API:**

To display the image and the livestream of the camera we need to sent request to Google Mirror API service to create new static card and add it to the timeline of google glass. Requests to the Google Mirror API must be authorized using OAuth 2.0 credentials. So, first we will get the credentials from the user.

```
private void fetchTokenForAccount(final String account) {
    // We fetch the token on a background thread otherwise Google Play
    // Services will throw an IllegalStateException
    sThreadPool.execute(new Runnable() {
        @Override
        public void run() {
            try {
                // If this returns immediately the OAuth framework thinks
                // the token should be usable
                final String token = GoogleAuthUtil.getToken(
                        LocationStatus.this, account, SCOPE);

                if (token != null) {
                    // Pass the token back to the UI thread
                    Log.i(TAG, String.format("getToken returned token %s", token));
                    mHandler.post(new Runnable() {
                        @Override
                        public void run() {
                            onTokenResult(token);
                        }
                    });
                }
            } catch (final UserRecoverableAuthException e) {
                // This means that the app hasn't been authorized by the user for access
                // to the scope, so we're going to have to fire off the (provided) Intent
                // to arrange for that. But we only want to do this once. Multiple
                // attempts probably mean the user said no.
                Log.i(TAG, "Handling a UserRecoverableAuthException");

                mHandler.post(new Runnable() {
                    @Override
                    public void run() {
                        startActivityForResult(e.getIntent(), REQUEST_AUTHORIZATION);
                    }
                });
            } catch (IOException e) {
                // Something is stressed out; the auth servers are by definition
                // high-traffic and you can't count on 100% success. But it would be
                // bad to retry instantly, so back off
                Log.e(TAG, "Failed to fetch auth token!", e);

                mHandler.post(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(LocationStatus.this,
                                "Failed to fetch token, try again later", Toast.LENGTH_LONG).show();
                    }
                });
            } catch (GoogleAuthException e) {
                // Can't recover from this!
                Log.e(TAG, "Failed to fetch auth token!", e);

                mHandler.post(new Runnable() {
                    @Override
                    public void run() {
```

```
                    Toast.makeText(LocationStatus.this,
                            "Failed to fetch token, can't recover", Toast.LENGTH_LONG).show();
                }
            });
        }
    }
});
}
```

Then we used createNewTimelineItem to send the image and livestream to google glass

```
private void createNewTimelineItem() {
    if (!TextUtils.isEmpty(mAuthToken)) {

        Mirror service = new Mirror.Builder(new NetHttpTransport(), new AndroidJsonFactory(), null)
                .setApplicationName(appName).build();

        try {
            URL url = new URL("http://192.168.8.101:88/cgi-
bin/CGIProxy.fcgi?cmd=snapPicture2&usr=glass2018&pwd=glass2018&t=");
            attachmentIS = url.openStream();
            //get content type
            HttpURLConnection connection = (HttpURLConnection)  url.openConnection();
            connection.setRequestMethod("HEAD");
            connection.connect();
            contentType = "image/jpeg";
            Log.d("contentType", contentType);
        } catch (MalformedURLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        InputStreamContent mediaContent = new InputStreamContent(contentType, attachmentIS);

        //Add Menu Items
        List<MenuItem> menuItemList = new ArrayList<MenuItem>();
        menuItemList.add(new MenuItem().setAction("DELETE"));
        menuItemList.add(new
MenuItem().setAction("OPEN_URI").setPayload("https://gglass2018.000webhostapp.com/Cam%20(2).html"));

        List<MenuValue> menuValues = new ArrayList<MenuValue>();
        menuValues.add(new MenuValue().setIconUrl("http://192.168.8.101:88/cgi-
bin/CGIProxy.fcgi?cmd=snapPicture2&usr=glass2018&pwd=glass2018&t=")
                .setDisplayName("Drill In"));
        menuItemList.add(new MenuItem().setValues(menuValues).setId("Refresh").setAction("CUSTOM"));

        //Create Timeline Item
        TimelineItem timelineItem = new TimelineItem();
        timelineItem.setNotification(new NotificationConfig().setLevel("DEFAULT"));
        timelineItem.setText("Room# "+ getIntent().getStringExtra("RoomNo"));
        timelineItem.setMenuItems(menuItemList);
        timelineItem.setId(itemId);


        try{
            if (contentType != null && contentType.length() > 0) {
                service.timeline().insert(timelineItem, mediaContent).setOauthToken((mAuthToken)).execute();
                //service.timeline().insert(timelineItem).setOauthToken((String) params[0]).execute();
```

```
            //service.timeline().attachments().insert(itemId, mediaContent).setOauthToken((String)
params[0]).execute();
            Toast.makeText(this, "Alarm was sent to google glass",
                  Toast.LENGTH_LONG).show();              }
      }
      catch (Exception e) {
         Log.d("Failure", e.getMessage());
         Toast.makeText(this, "Failed",
               Toast.LENGTH_LONG).show();
      }



   } else {
      Toast.makeText(this, "Sorry, can't create a new timeline card without a token",
            Toast.LENGTH_LONG).show();
   }
}
```

*Figure 50:Google glass*

# System Integration:

At the beginning of the project we started coding every component separately, which will make it easier to test and find any errors that might occurs. Then we combine the components for the guard. Finally, we started integrating all the components together until all of them is combined together.

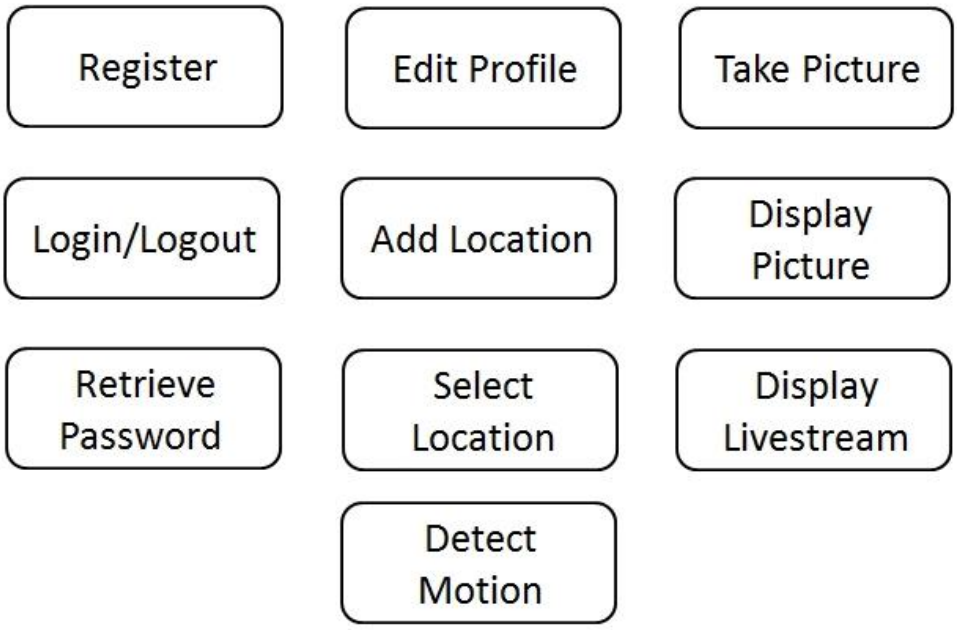



*Figure 51:stage 1 of Integration*

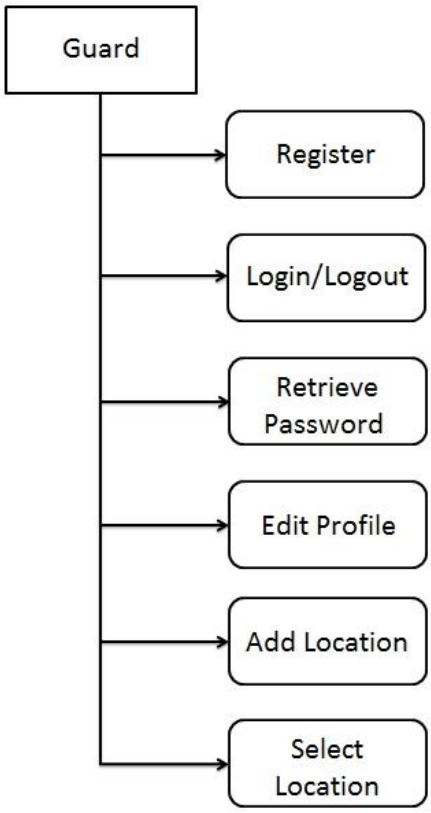



*Figure 52:stage 2 of Integration*

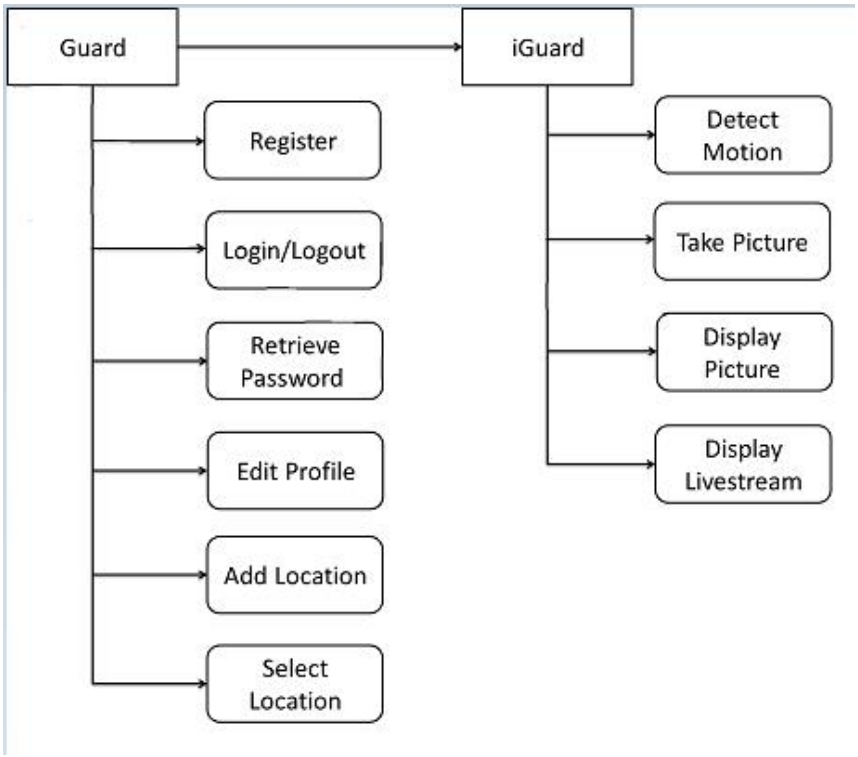



*Figure 53:stage 3 of Integration*

# CHAPTER 6: TESTING

# UNIT TESTING

In unit testing individual units/components of the system is tested individually to guarantee that it works as expected without any mistakes.

| Component name | Component description | Result |
|---|---|---|
| **Component A** | Register | Pass |
| **Component B** | Retrieve password | Pass |
| **Component C** | Log in | Pass |
| **Component D** | Edit Account | Pass |
| **Component E** | Add Location | Pass |
| **Component F** | Select Location | Pass |
| **Component G** | Authorization for mirror API | Pass |
| **Component H** | Connect to mirror API | Pass |
| **Component I** | Connect to Arduino | Pass |
| **Component J** | Start monitoring | Pass |
| **Component K** | Listen to the data send from the Arduino | Pass |
| **Component L** | Take picture from IP camera | Pass |
| **Component M** | Display picture on Google glass | Pass |
| **Component N** | Display live stream on Google glass | Pass |
| **Component O** | Stop monitoring | Pass |
| **Component P** | Sign out | Pass |

*Table 12:unit testing*

# INTEGRATION AND REGRESSION TESTING

### Integration Testing

In integration testing individual units are combined and tested as a group to assure that it does not affect and to expose faults in the interaction between integrated units.

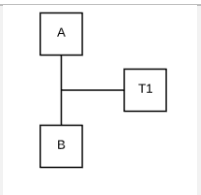
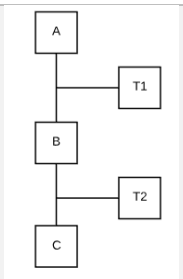
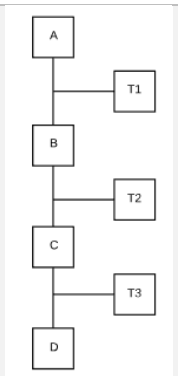
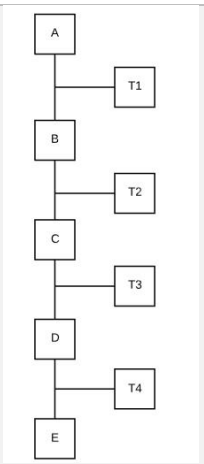
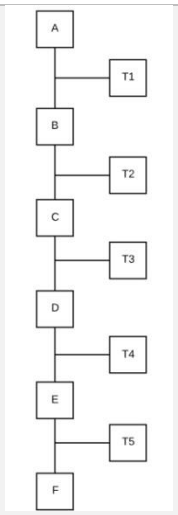
| Test Sequence | | | | | |
|---|---|---|---|---|---|
| **Test number** | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 |
| **Components** | Integrating Components A & B | Integrating Components C , Test 1 | Integrating Components D , Test 2 | Integrating Components E , Test 3 | Integrating Components F , Test 4 |
| **Graph** |  |  |  |  |  |
| **Description** | Register and Retrieve Password | Connect test 1 with Login | Connect test 2 with Edit account | Connect test 3 with Add Location | Connect test 6 with Select Location |
| **Result** | Pass | Pass | Pass | Pass | Pass |

*Table 13:integration testing1*

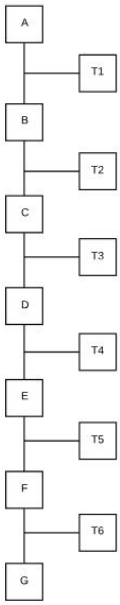
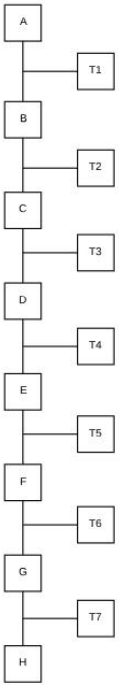
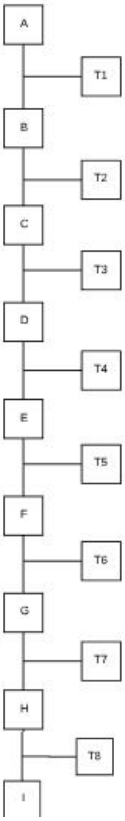
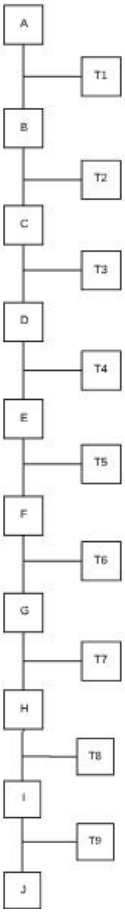
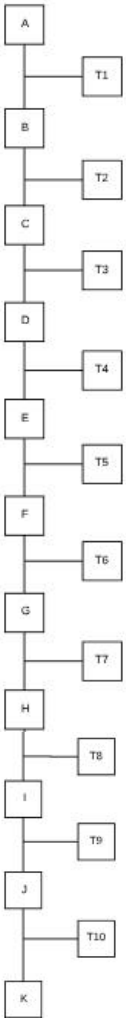
| Test number | Test 6 | Test 7 | Test 8 | Test 9 | Test 10 |
|---|---|---|---|---|---|
| **Components** | Integrating Components G, Test 5 | Integrating Components H , Test 6 | Integrating Components I, Test 7 | Integrating Components J, Test 8 | Integrating Components K , Test 9 |
| **Graph** |  |  |  |  |  |
| **Description** | Connect test 5 with Authorization for mirror API | Connect test 6 with Connect to mirror API | Connect test 7 with Connect to Arduino | Connect test 8 with Start monitoring | Connect test 9 with Listen to the data send from the Arduino |
| **Result** | Pass | Pass | Pass | Pass | Pass |

*Table 14: integration testing 2*

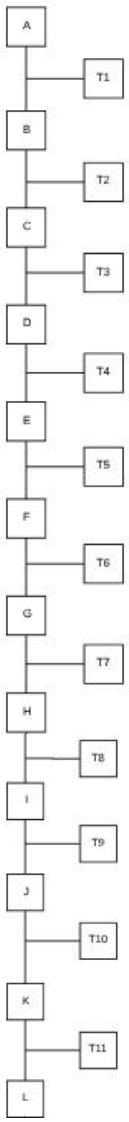
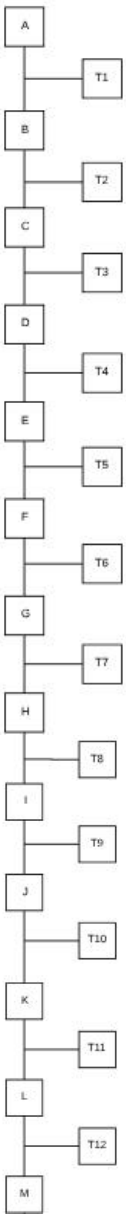
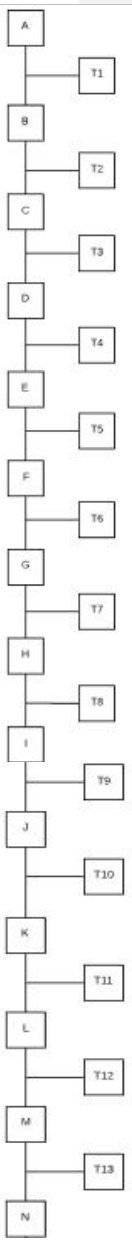
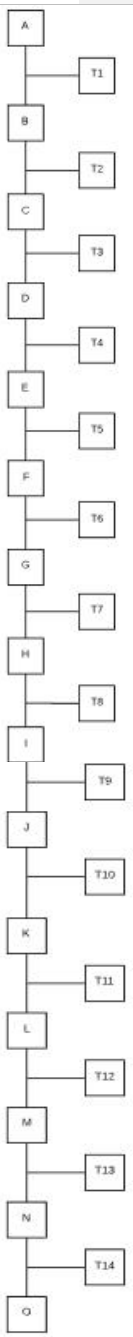
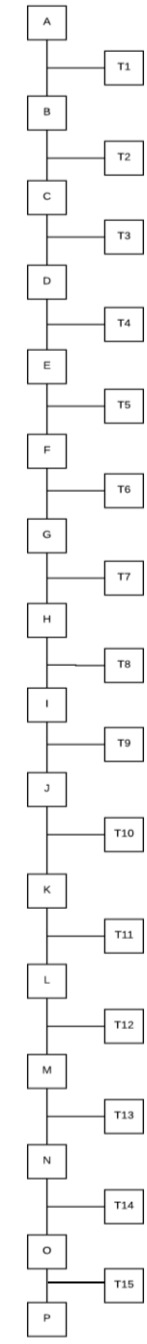
| Test number | Test 11 | Test 12 | Test 13 | Test 14 | Test 15 |
|---|---|---|---|---|---|
| **Components** | Integrating Components L , Test 10 | Integrating Components M, Test 11 | Integrating Components N, Test 12 | Integrating Components O, Test 13 | Integrating Components P, Test 14 |
| **Graph** |  |  |  |  |  |
| **Description** | Connect test 10 with Take picture from IP camera | Connect test 11 with Display picture on Google glass | Connect test 12 with Display live stream on Google glass | Connect test 13 with Stop monitoring | Connect test 14 with Sign Out |
| **Result** | Pass | Pass | Pass | Pass | Pass |

*Table 15:integration testing 3*

**Regression Testing**

Regression testing is done to ensures that that the system is still perform the same way after it is changed or modification is done.

| Action | Expected output | System output | True match | False match |
|---|---|---|---|---|
| **Add new address location** | The app should add the location successfully | The location added | √ | |
| **Press start monitoring button** | The Arduino start sensing for motion detection | Success notification appear when sensing motion | √ | |
| **Capture image** | The camera capture image when sensing motion | The system sends the image to the google glass | √ | |
| **Display image** | The image is display in google glass with the date was taken | The image displayed successfully | √ | |

*Table 16:Regression Testing*

# PERFORMANCE AND STRESS TESTING:

In this section of testing we test each function separately for 3 times to ensure software applications will perform well under their expected workload.

| Action | Expected output | System output | True match | False match |
|---|---|---|---|---|
| App Start Up: **the user taps on app icon** | The home page should be shown in 1-2 seconds | The app start in 2 seconds | √ | |
| Data to and from Firebase cloud: **The app should handle data efficiently that is sent from the Firebase cloud** | The app must not take too much time while loading data | The app retrieves the data efficiently | √ | |
| Network Performance: **the user taps on location address** | Quick response with Success notification | The app responding with Success notification | √ | |
| Network Speed: **test the app on Wi-Fi networks.** | The app should work very well. | The app work on different networks. | √ | |

*Table 17:PERFORMANCE AND STRESS TESTING*

We use to test the performance of the application the android studio IDE, android profiler which is built-in tool that shows the CPU, memory and network performance:

This figure shows the CPU, memory and network states when Start monitoring:

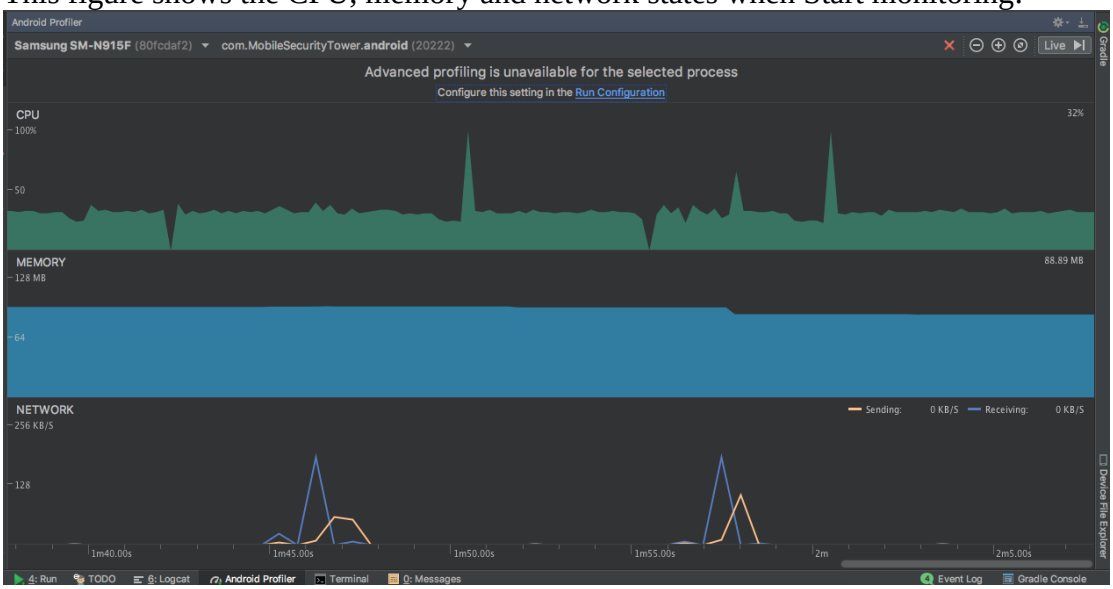



*Figure 54:CPU, memory and network states when Start monitoring*

This figure shows the CPU, memory and network states when selecting a location, the sensors detects a motion and take a picture:

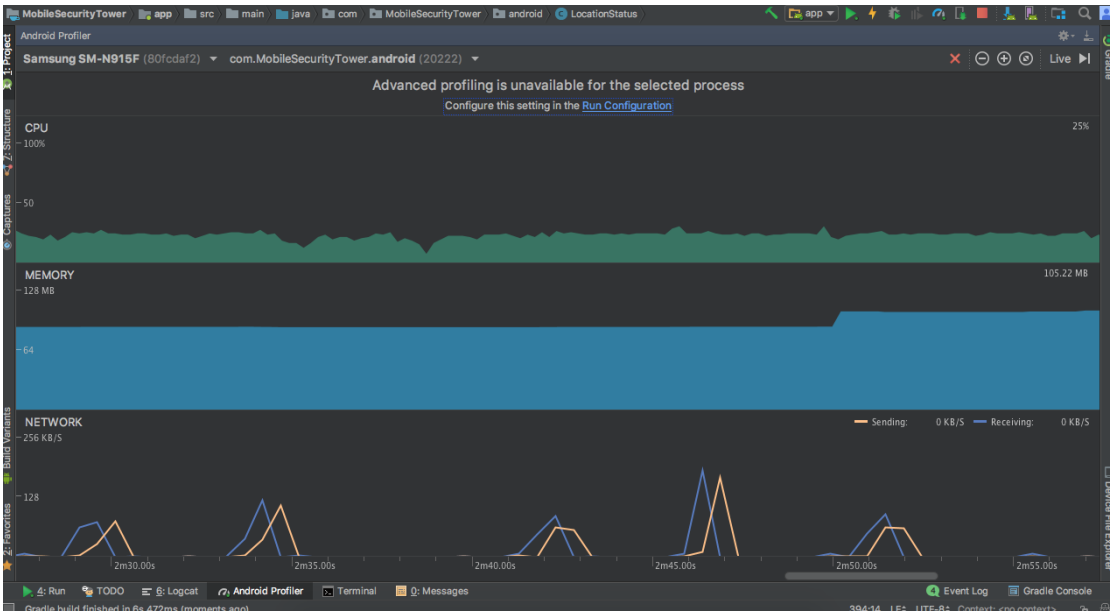



*Figure 55:CPU, memory and network states when selecting a location*

This figure shows the CPU, memory and network states the system while open the application without performing any function:

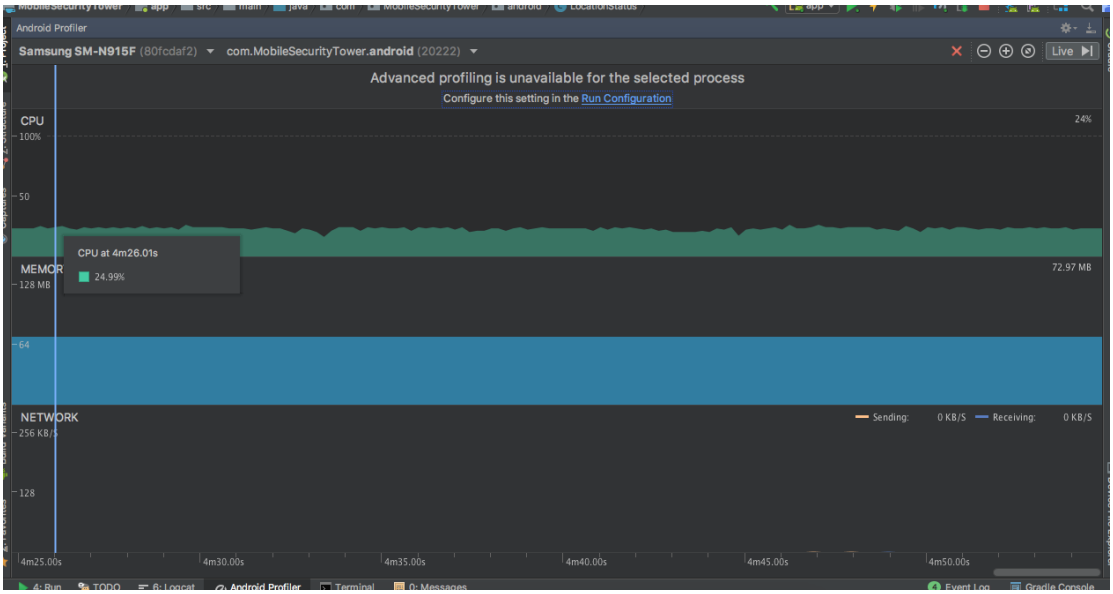



*Figure 56:CPU, memory and network states the system while open the application without performing any function*

# USER ACCEPTANCE TESTING

In user acceptance testing we asked group of security guards to follow a scenario

We create a survey after they finish using the system our survey was published using google forms. Survey is the best way to get users feedback since it is anonymous google forms arranged the results in pie chart and we analyses the results.

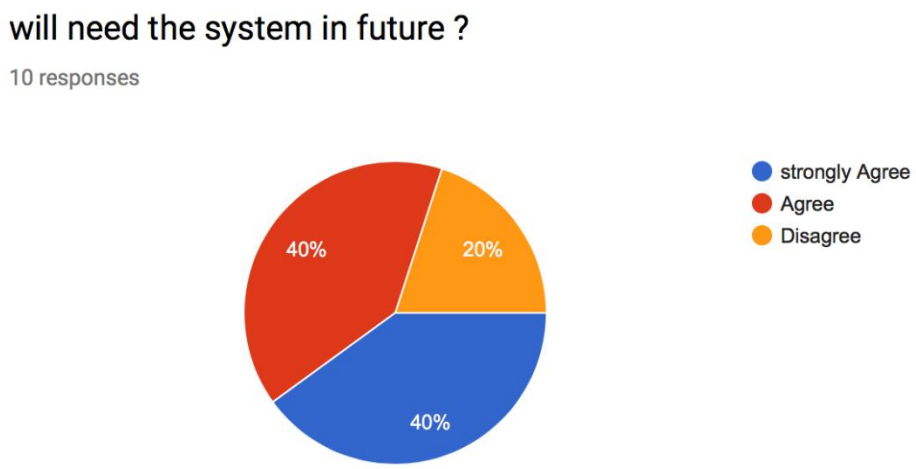



*Figure 57: User Acceptance  Surevy 1*

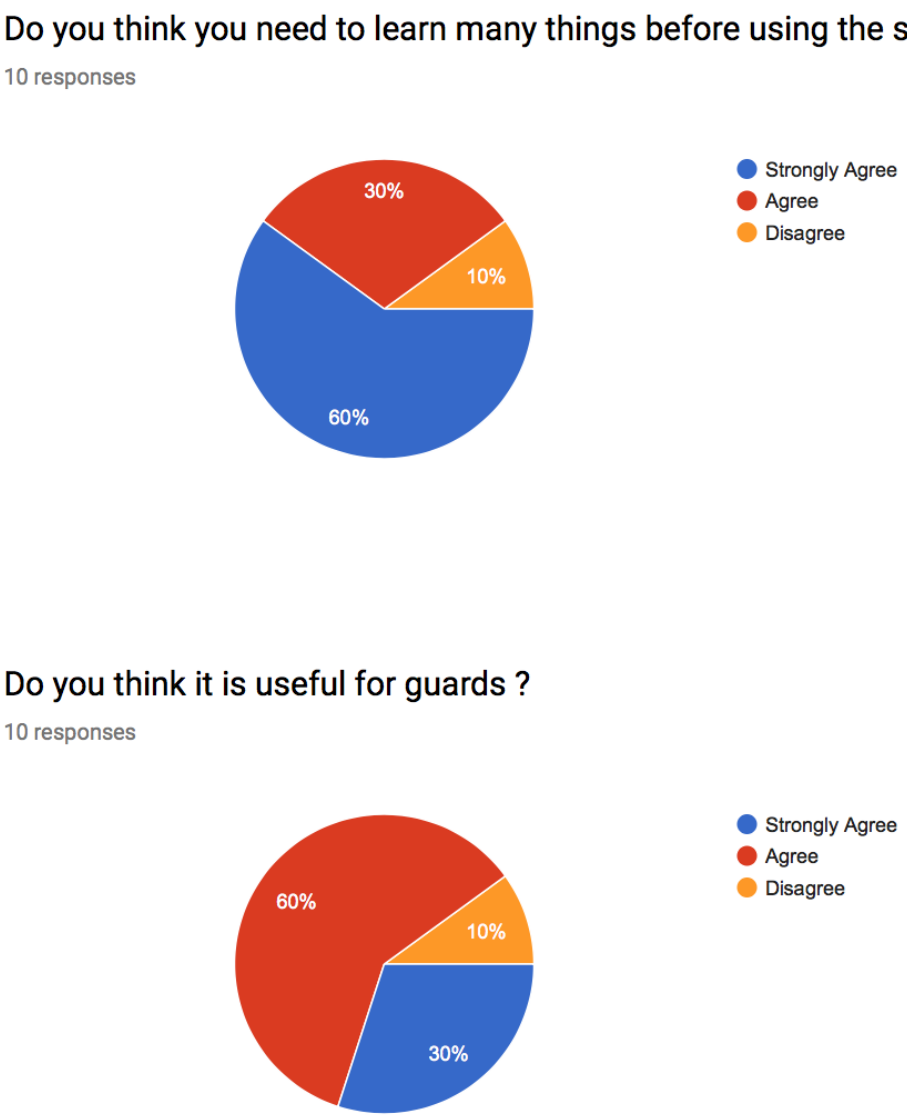



*Figure 58:User Acceptance  Survey 2*

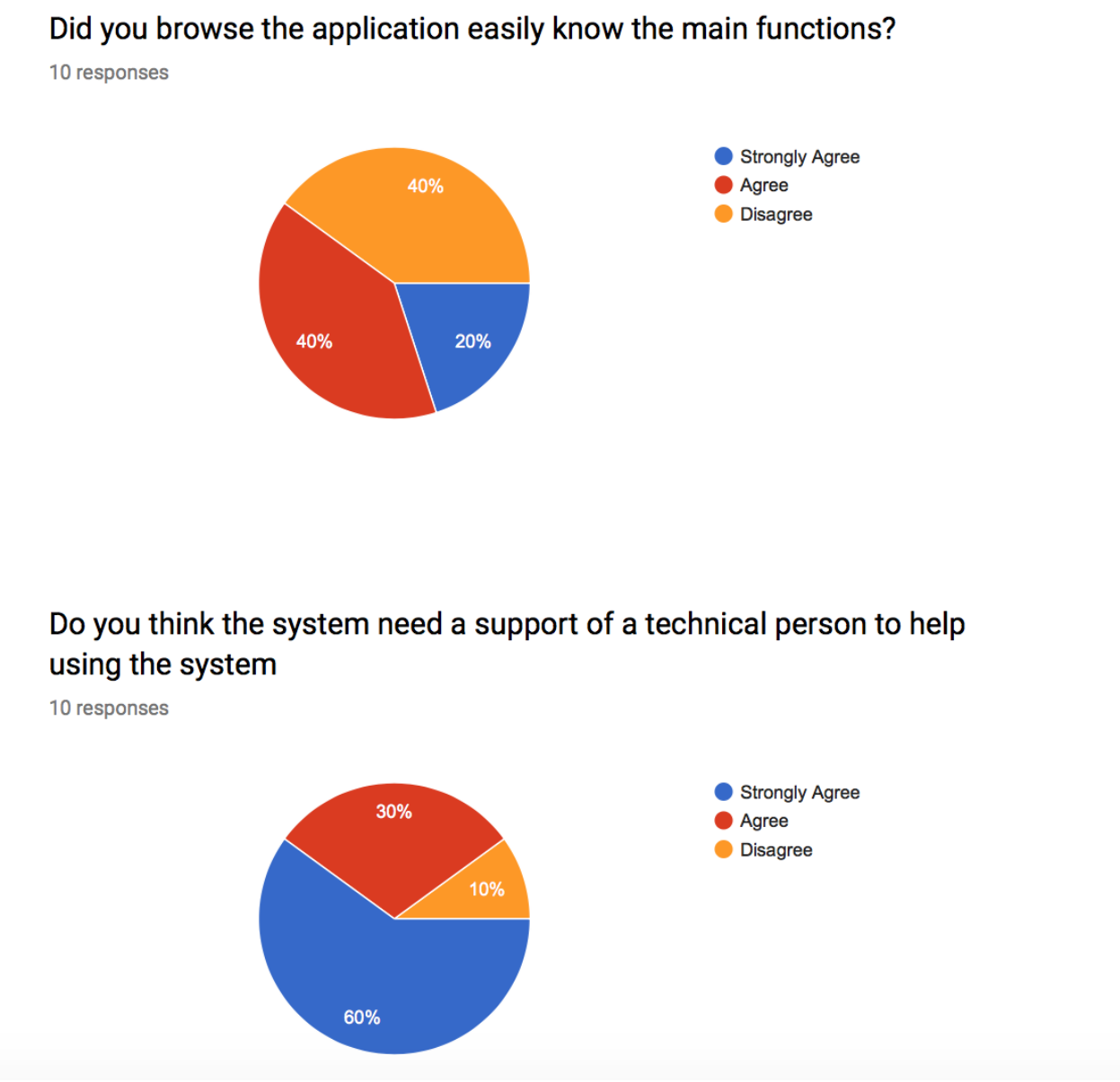



*Figure 59:User Acceptance  Survey 3*

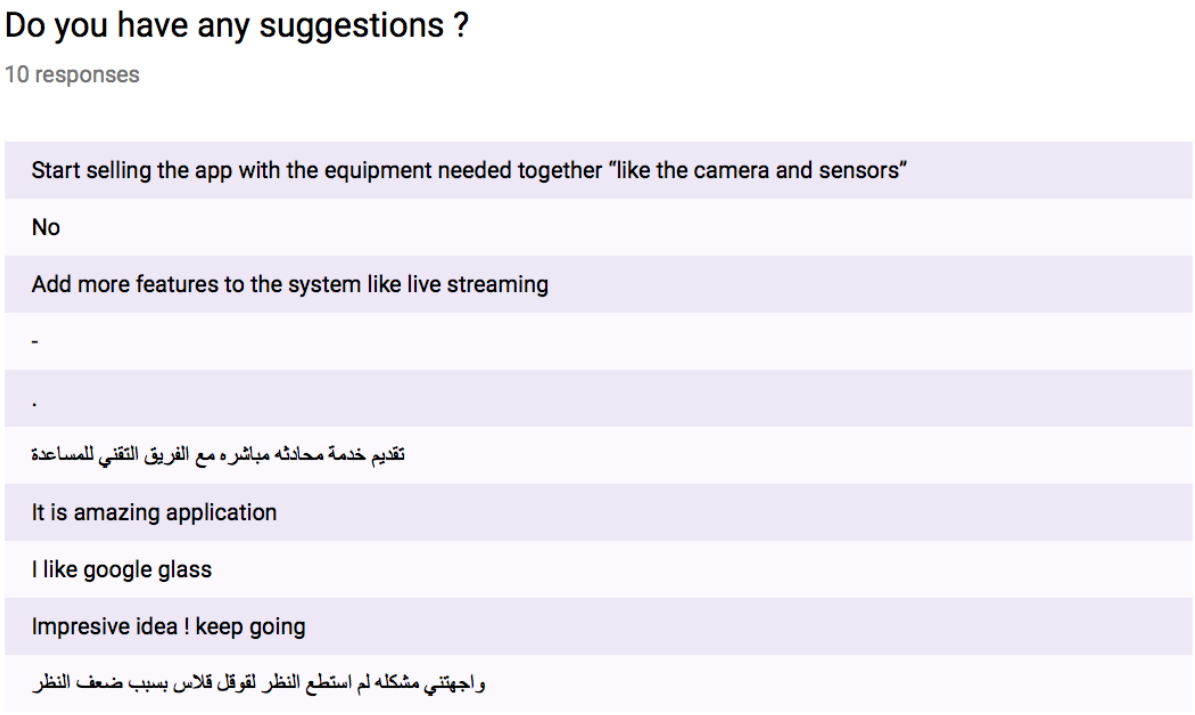



*Figure 60: User Acceptance Survey 4*

After analyzing the results:

- Overall impression was good .
- Some users found difficulties to use the user without support .
- The users need to learn how to use the hardware of the system
- More features need to be added to the system in the future such as : live chatting with technical support.

| # | Task: |
|---|---|
| **1** | Sign up |
| **2** | Restore your password |
| **3** | Login |
| **4** | Add location |
| **5** | Edit profile |
| **6** | Select location |
| **7** | Start monitor |
| **8** | Stop monitoring |
| **9** | Log out |

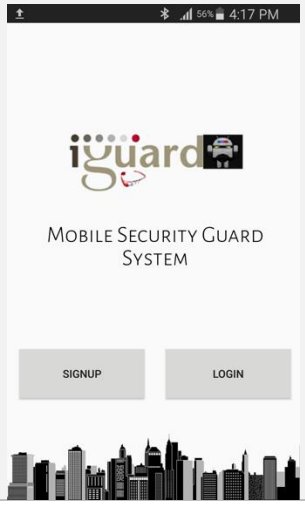
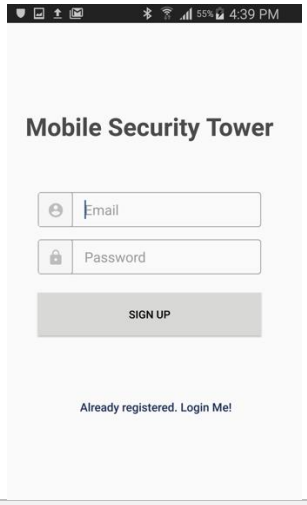
*Figure 61:Task for User Acceptance*

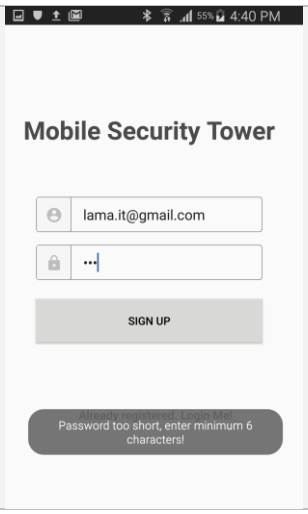
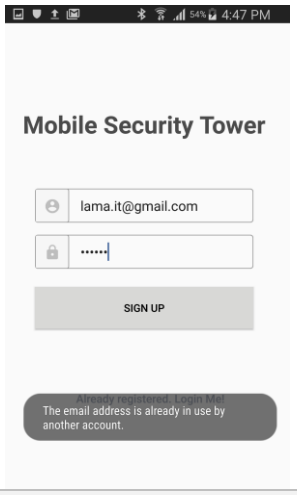
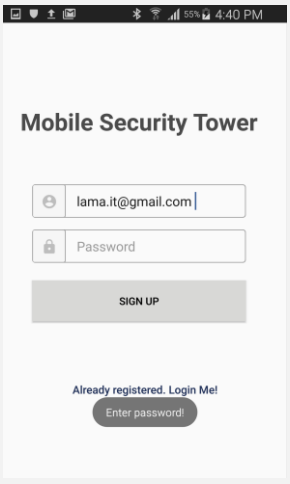
# TEST CASES

In the test cases, we test each function with all possible actions (update or changes, output) that may occur. To make sure that the system is reliable and function as expected.

## 1.Signup:

Function: signup the user in the system
Description: test how can the user signup
Input data: email, password

| Test case# 1 | | |
|---|---|---|
| **Description** | General setting | |
| **Test case description** | Expected output | Actual result (pass/failed) |
| **The user will click on application icon** | The application will start and home page will open | Pass   |
| **The user click signup button** | Signup page will be displayed | Pass   |
| **The user click signup button with a short password Username:lama.igurad@gmail.com Password:123** | Warning massage will appear that the password to short | Pass |

| | | |
|---|---|---|
| | |  |
| **The user click signup button with an email already used** | Warning massage will appear that the email already exists | Pass<br>  |
| **The user click signup without typing password** | Warning massage will appear password is required | Pass<br>  |

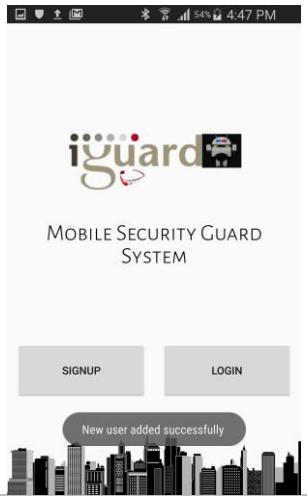
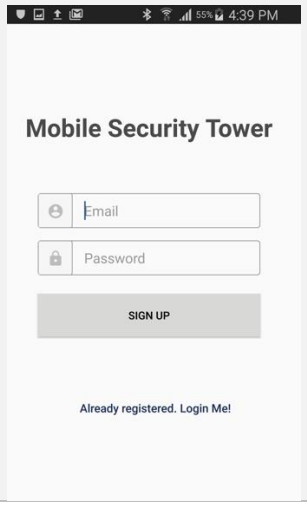
| | | |
|---|---|---|
| **The user enter email , password and click on signup** | Confirmation message appears indicate that a new user added successfully. | Pass   |

*Figure 62:Test case 1*

## 2.Login:

Function: Login into system
Description: Test how can the user login
Input Data: Email, password

| **Test case# 2** | | |
|---|---|---|
| **Description** | Login | |
| **Test case Description** | Expected Output | Actual Result (pass/failed) |
| **The user click login button** | Login page will be displayed | Pass   |
| **The user enter a wrong Email or password** | Error massage will appear to tell the user if he enter a wrong password or he did not signup | Pass |

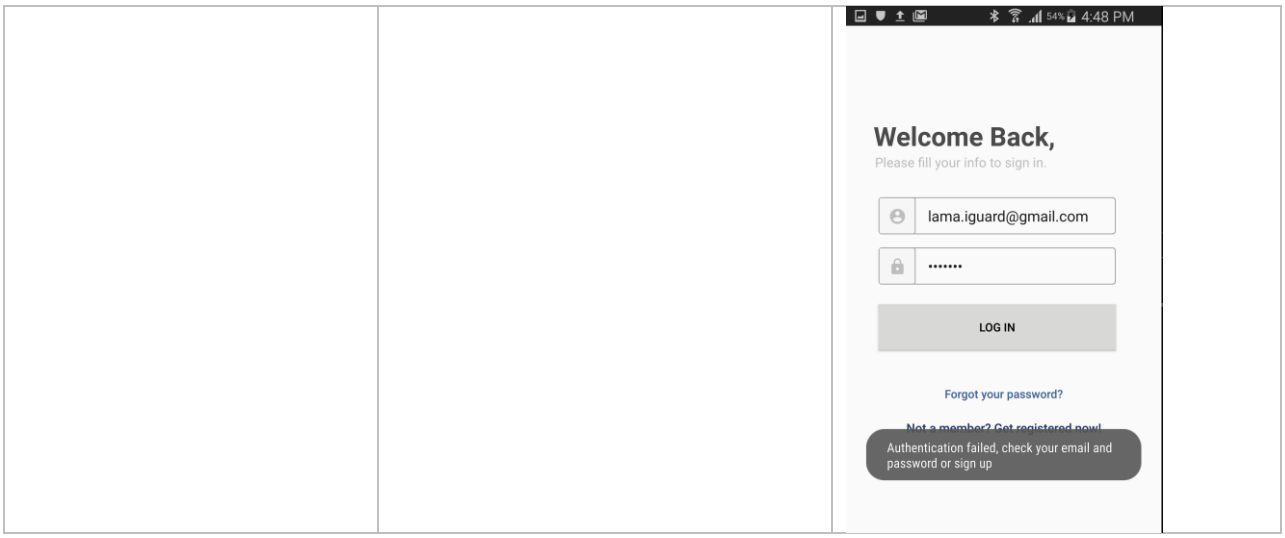



*Figure 63:Test case 2*

## 3.Retrieve password:

Function: retrieve password
Description: test how can the system retrieve the password
Input data: email

| **Test case# 3** | | |
|---|---|---|
| **Description** | Retrieve password | |
| **Test case Description** | Expected Output | Actual Result (pass/failed) |
| **The user click forget password and enter a wrong email** | error massage will appear the email does not exist | Pass |

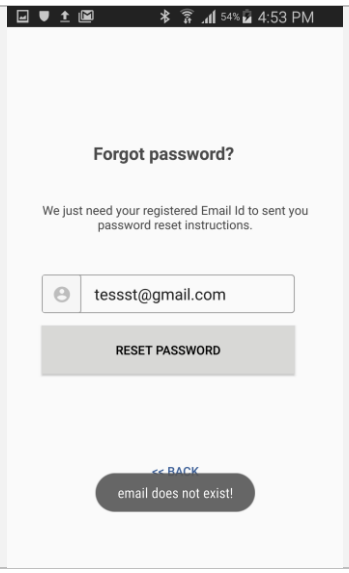
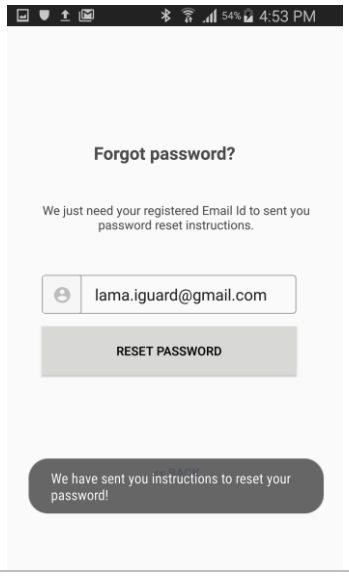
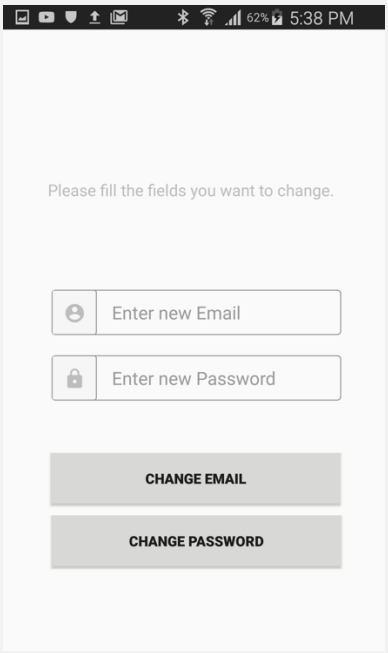
| | | |
|---|---|---|
| | |  |
| **The user click forgot the password and enter a right email** | Confirmation message appear | Pass<br>  |

*Figure 64:Test case 3*

## 4.Edit profile:

Function: Edit profile
Description: Test how can the user edit profile
Input Data: Email, password

| Test case# 4 | | |
|---|---|---|
| **Description** | General setting | |
| **Test case Description** | Expected Output | Actual Result (pass/failed) |
| **The user will click on Edit profile** | The application will open edit profile page | Pass   |
| **The user clicks change email without filling the new email field** | Warning message appear | Pass |

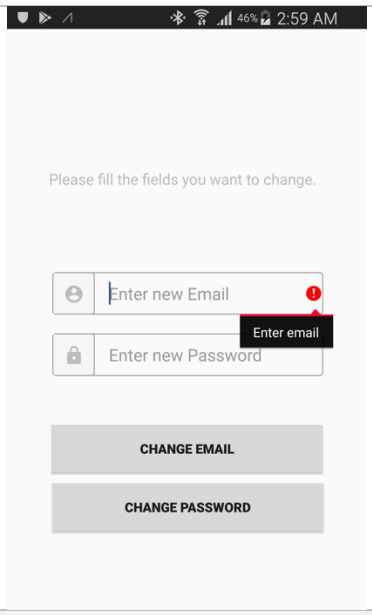
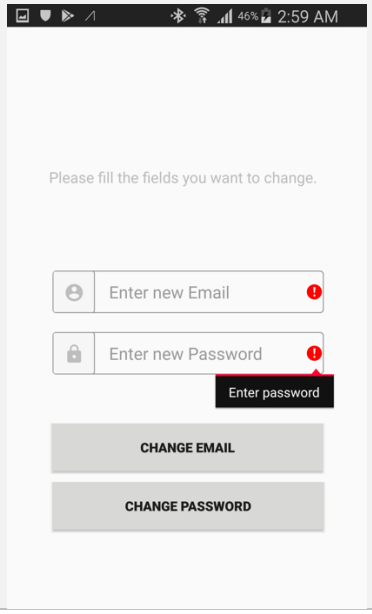
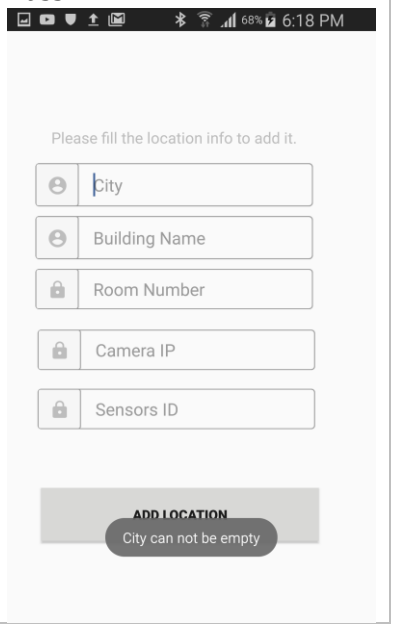
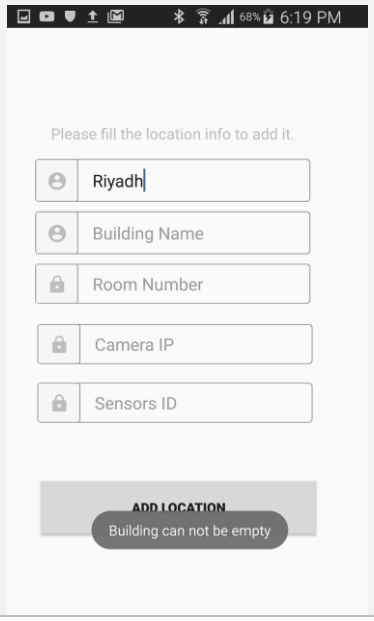
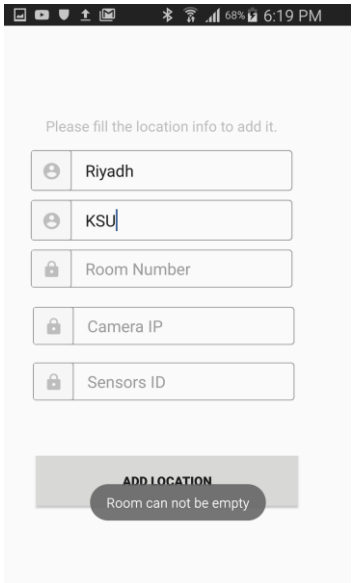
| | | |
|---|---|---|
| | |  |
| **The user clicks change password without filling the new password field** | Warning message appear | Pass<br>  |

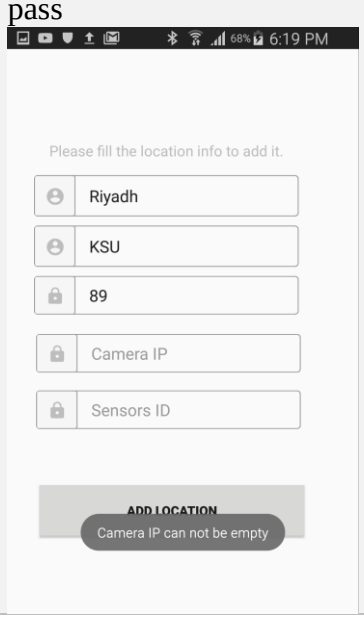
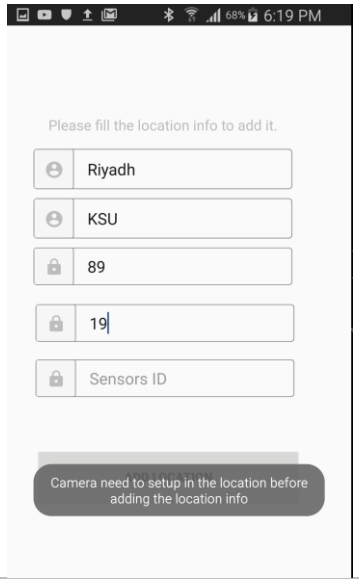
*Figure 65:Test case 4*

## 5.Add location:

Function: add location to the system
Description: test how can the user add location
Input data: city, building name, room number, camera ip, sensors id

| Test case# 5 | | |
|---|---|---|
| **Description** | Add location | |
| **Test case Description** | Expected Output | Actual Result (pass/failed) |
| **The user will click on Add location** | The application will open Add location page | Pass |
| **The user does not type the city name** | Warning message appear | Pass  |

| | | |
|---|---|---|
| **The user does not type the building name** | Warning message appear | Pass<br>  |
| **The user does not type room number** | Warning message appear | Pass<br>  |

| | | |
|---|---|---|
| **The user does not type the camera IP** | Warning message appear | pass   |
| **The user does not setup the camera** | Warning message appear | Pass   |

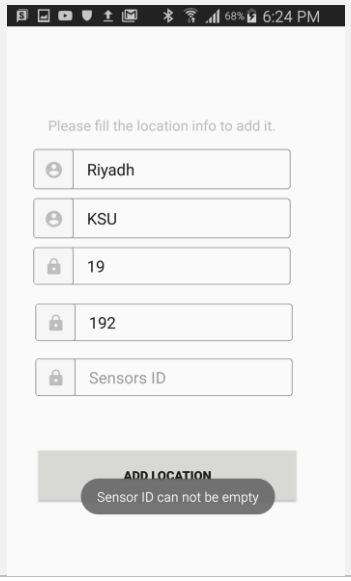
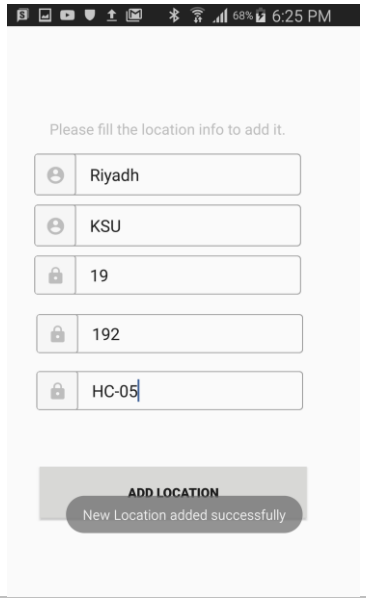
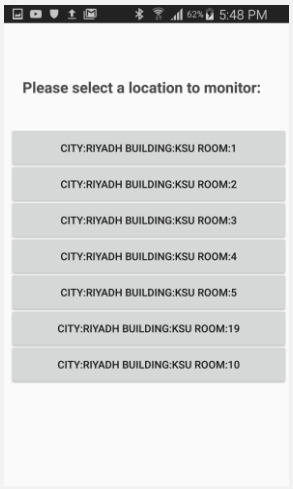
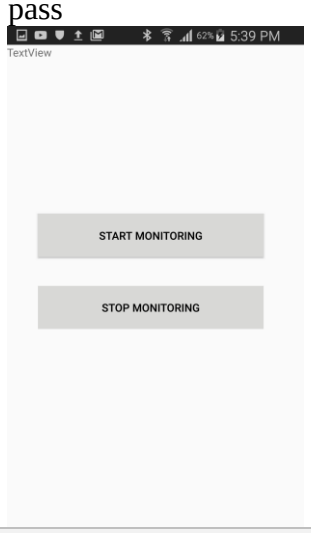
| | | |
|---|---|---|
| **The user does not enter the sensor ID** | Warning message appear | Pass   |
| **The user enters all correct information and setup the sensors and camera and press add location** | Confirmation message appear | Pass   |

*Figure 66:Test case 5*

## 6.Select location:

Function: select location to the monitor
Description: test how can the user select location

| Test case# 6 | | |
|---|---|---|
| **Description** | Select location | |
| **Test case Description** | Expected Output | Actual Result (pass/failed) |
| **The user click on select location button** | List of locations is displayed | Pass   |
| **After user select one of the locations** | it will open this page the user can select start or stop monitoring | pass   |
| **Authentication for google glass** | List of account is displayed | pass |

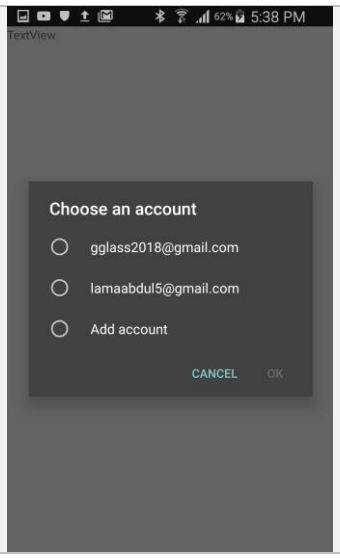
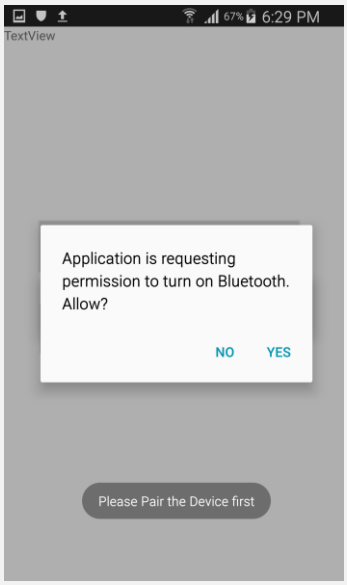
| | | |
|---|---|---|
| | |  |

*Figure 67:Test case 6*

## 7. Detect motion:

Function: sense any motion

Description: test how can the Arduino sense motion detection.

| Test Case# 7 | | |
|---|---|---|
| **Description** | **Detect motion** | |
| **Test Case Description** | Expected Output | Actual Result (Pass/Failed) |
| **After User Select Start Monitoring** | The Application Will Request From User To Turn On Bluetooth To Start Detecting The Motion | Pass  |
| **If The Sensor Detects Any Motion** | Sensor Tell There Is Motion Detected | Pass |

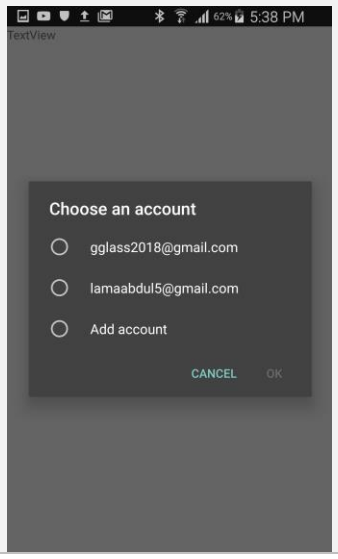
| | | |
|---|---|---|
| | |  |
| **If The System Does Not Detects Any Motion** | Sensors Does Not Send Any Thing | Pass |

*Figure 68: Test case 7*

## 8.Display image on Google Glass:

Function: display picture on Google Glass

Description: Test how can the system display the captured image on google glass

| **Test case# 7** | | |
|---|---|---|
| **Description** | **Display image on Google Glass** | |
| **Test case Description** | Expected Output | Actual Result (pass/failed) |
| **Ask the user to select the email that is used in google glass.** | List of google account is displayed to choose one of them. | Pass  |

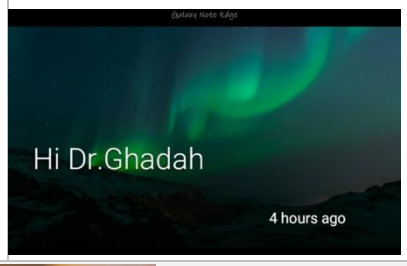
| | | |
|---|---|---|
| **After the camera take image it will be sent to google glass** | Image is displayed in google glass | pass   |
| |  | |

*Figure 69:Test case 8*

# CHAPTER 7: CONCLUSION

# CONCLUSION:

In the context of our graduation project, we designed the iGuard system which will add mobility to guards to improve the current state of security devices by implementing the idea of using Google Glass as a mobile security Guard.

Project goals and objectives, and the problem and the proposed solution are achieved. In this document, the used technologies and techniques are presented. The most prominent applications in the field are reviewed in order to highlight the related functionalities. Then we presented a comparison table of some reviewed applications to emphasize the added value of our project. In Chapter 4 we wrote the functional and non-functional requirements, as well as the following diagrams: use case diagram that shows the main functions of the system, sequence diagrams, conceptual diagram and class diagram. Chapter 5 illustrates the system architecture and the interface designs. And then present the implementation and integration of the system in Chapter 6, chapter 7 is testing the system. Finally, we concluded.

## Global And Local Impact

iGuard is supposed to improve the current state of security management in important, big and sensitive facilities that will ensure the security at the building of any unauthorized access or especially out of working hours. One of the most functionalities provided in our system is to let the guards keep their mobility and give them the ability to move and monitor at the same time. Using iGuard empowers surveillance systems with efficient cost effective and less human resources. In addition, iGuard would improve decision making and speeds up action towards violence and unauthorized access to facilities.

## Limitations

Technical limitation comprises the data storage and battery of the Google glass for long time performance.

## Future Work

- Develop an IOS version
- Add Arabic language to the application