Home Defense System

Software Requirements Specification

1.0

9/1/15

Submitted to

Dr. Razib Iqbal

Assistant Professor of Computer Science

Missouri State University

in partial fulfillment of the

CSC450 Course Project Deliverables

Fall 2015

**Team 5**

Ian E O’Dell

[lan0@live.missouristate.edu](mailto:lan0@live.missouristate.edu)

Christian Kirk [Christian42@live.missouristate.edu](mailto:Christian42@live.missouristate.edu)

Michael Mcbride [Mcbride555@live.missouristate.edu](mailto:Mcbride555@live.missouristate.edu)

Coury Ferson

[Ferson7@live.missouristate.edu](mailto:Ferson7@live.missouristate.edu)

Tianqi Huang

Tianqi92@live.missouristate.edu

# Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Description** | **Comments** |
| 9/1/15 | Version 1.0 | Drafting process |
|  |  |  |
|  |  |  |
|  |  |  |

# Document Approval

The following Software Requirements Specification has been accepted and approved by the following stakeholders

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Ian O’Dell | Technical Writer |  |
|  | Tianqi Huang | Interim Project Manager |  |
|  | Dr. R. IQBAL | Instructor/Customer |  |

**Table of Contents**

Revision History ii

Document Approval ii

1. Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 1

1.4 References 1

1.5 Overview 1

2. General Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Characteristics 2

2.4 General Constraints 2

2.5 Assumptions and Dependencies 2

3. Specific Requirements 2

3.1 External Interface Requirements 3

3.1.1 User Interfaces 3

3.1.2 Hardware Interfaces 3

3.1.3 Software Interfaces 3

3.1.4 Communications Interfaces 3

3.2 Functional Requirements 3

3.2.1 <Functional Requirement or Feature #1> 3

3.2.2 <Functional Requirement or Feature #2> 3

3.3 Use Cases 3

3.3.1 Use Case #1 3

3.3.2 Use Case #2 3

3.4 Classes / Objects 3

3.4.1 <Class / Object #1> 3

3.4.2 <Class / Object #2> 3

3.5 Non-Functional Requirements 4

3.5.1 Performance 4

3.5.2 Reliability 4

3.5.3 Availability 4

3.5.4 Security 4

3.5.5 Maintainability 4

3.5.6 Portability 4

3.6 Inverse Requirements 4

3.7 Design Constraints 4

3.8 Logical Database Requirements 4

3.9 Other Requirements 4

4. Analysis Models 4

4.1 Sequence Diagrams 5

4.3 Data Flow Diagrams (DFD) 5

4.2 State-Transition Diagrams (STD) 5

5. Change Management Process 5

A. Appendices 5

A.1 Appendix 1 5

A.2 Appendix 2 5

# 1. Introduction

Home security can always be improved upon. This product is that next step to securing your home and making you as safe as possible. Our product is affordable, highly documented, and easy to use. What’s currently out there now is overcomplicated and doesn’t do everything that it should. For example, Panasonic offers a Home Monitoring Kit for over $250, and it’s required you have a landline phone! People steal, houses burn down, and if you have children you need to keep them safe. With our Raspberry Pi Home Defense System (HDS) you’ll be protected, and it won’t break the bank.

## 1.1 Purpose

To protect homeowners from burglers and fires.

## 1.2 Scope

The Home Defense System (HDS) will detect motion and alert the user of the detection. The HDS also has a temperature gauge to detect extreme heat in case of fire, as well as a decibal monitor that alerts the user when a very loud noise (e.g. gunshot, glass break) is identified.

## 1.3 Definitions, Acronyms, and Abbreviations

HDS – Home Defense System

## 1.4 References

Panasonic.com

9/2/15

<http://shop.panasonic.com/home-and-office/connected-home/connected-home-complete-systems/KX-HN6002W.html>

ADT.com

9/4/15

http://new.adt.com/?ecid=DCore2-Trad-060515

## 1.5 Overview

This document contains a general overview of our product, its functions, our intended user

market, constraints, and assumptions and dependencies. The section following the overiew will

go in to great detail about the specific requirements of the system and each of its constituent

parts. It is broken down into five sections: external interface requirements, functional

requirements, use cases, class and objects, and non-functional requirements.

# 2. General Description

Basically everyone has a place they call home, and we want the HDS to protect it while you are away or sleeping. It’s affordable, and it’s basic monitering systems will send alerts to your phone in case of an emergency.

## 2.1 Product Perspective

Unlike other home defense software the HDS is independent and totally self-contained. Compared to a leading competitor ADT, which is a product that is a component of a larger system because of the call centers and the expensive services ADT provides. What sets the HDS apart from others is it’s affordability and low maintanability.

## 2.2 Product Functions

* Connects door to raspberry pi to tell when the door is opening or getting broken into
* Receive Email/SMS alerts when an alarm occurs.
* You may monitor your home from your smart phone or computer using the dashboard
* Temperature is logged every 5 minutes with a timestamp to check if there is a fire, or to see if the oven was left on.
* User can record footage at specific times throughout the day e.g. 2:00p.m. - 3:00p.m.
* Motion detector that will enable the camera to begin recording.

## 2.3 User Characteristics

The user of this product should have at least a high school education, with basic knowledge of smart phone technology. No technical expertise is required.

## 2.4 General Constraints

In order for the system to function properly, it must be connected to Microsoft’s Azure cloud services via the internet.

## 2.5 Assumptions and Dependencies

The system will use Microsoft’s Azure cloud computing services for the policy engine computing. The system and cloud services will be programmed using Python.

# 3. Specific Requirements

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project’s software design, implementation, and testing.

Each requirement in this section should be:

* Correct
* Traceable (both forward and backward to prior/future artifacts)
* Unambiguous
* Verifiable (i.e., testable)
* Prioritized (with respect to importance and/or stability)
* Complete
* Consistent
* Uniquely identifiable (usually via numbering like 3.4.5.6)

Attention should be paid to the carefuly organize the requirements presented in this section so that they may easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.

## 3.1 External Interface Requirements

## 3.1.1 User Interfaces

## The user will have a mobile application to control when the alarm is armed/not armed. The user will also have the option to set a specific temperature that will notify the user when that temperature is reached.

## 3.1.2 Hardware Interfaces

## No hardware interface, the only hardware required is a smartphone and raspberry pi.

## 3.1.3 Software Interfaces

## The software will be written in Python code.

## 3.1.4 Communications Interfaces

## The cloud will retrieve information from the raspberry pi, then the cloud will transmit that information to the smartphone app. 3.2 Functional Requirements

## 3.2 Functional Requirements

### 3.2.1 Motion detection alarm system

3.2.1.1 Introduction

* The system shall alarm the user after the system detects any motion in the image which is captured by the systems camera.

3.2.1.2 Inputs

* The systems camera shall take video after the power is plugged in.
* The systems camera shall take photo after the power plugged in.
* The system shall transfer the recorded video to the cloud.
* The system shall transfer the recorded photo to the cloud.

3.2.1.3 Processing

* The system shall detect any motion found from the video in the cloud.

3.2.1.4 Outputs

* The system shall send the alarm to the user after the system detects motion.
* The system shall send the snapped video where the motion was detected to the user after the system detects motion.

3.2.1.5 Error Handling

* The system shall send an error to the supplier after an error occurred.
* The system shall stop working after an error occurred.

### 3.2.2 Temperature detection alarm system

3.2.2.1 Introduction

* The system shall alarm the user after the system detects a temperature over 150 degree in Fahrenheit which is captured by the systems temperature sensor.

3.2.2.2 Inputs

* The systems temperature sensor shall detect temperature after the power is plugged in.
* The system shall transfer the recorded temperature data to the cloud.

3.2.2.3 Processing

* The system shall determine the temperature: whether its higher than 150 degree in Fahrenheit in the cloud.

3.2.2.4 Outputs

* The system shall send an alarm to the user after the system detects a temperature higher than 150 degrees in Fahrenheit.

3.2.2.5 Error Handling

* The system shall send an error to the supplier after an error occur.
* The system shall stop working after an error occur.

## 3.3 Use Cases

### 3.3.1 Use Case #1

### 3.3.2 Use Case #2

…

## 3.4 Classes / Objects

### 3.4.1 <Class / Object #1>

3.4.1.1 Attributes

3.4.1.2 Functions

<Reference to functional requirements and/or use cases>

### 3.4.2 <Class / Object #2>

…

## 3.5 Non-Functional Requirements

### 3.5.1 Performance

* The product shall take no longer than 10 seconds to process the incoming data through the policy engine.
* The product shall provide no less than 360P quality of imagery.
* The product shall take no longer than 10 seconds to boot up the system.
* The product shall take no longer than 10 seconds to alarm the user after the sensor detects motion.

### 3.5.2 Reliability

* The product shall work 24 hours per day, 365 days per year.
* The product shall work under both power and internet plug in.
* The product shall work without human attention.

### 3.5.3 Availability

* The product shall allow the user to modify the policy through an app on their smart phone.
* The product shall be provided through a GUI.
* The product shall allow the user to turn on/off the system.

### 3.5.4 Security

* The product shall only allow the user to turn on/off the system.
* The product shall only send data to the raspberry pi which the user will then confirm.
* The product shall only allow the product provider to modify the software code.

### 3.5.5 Maintainability

* The product shall only allow the product provider to maintain the products integrity.

### 3.5.6 Portability

* The product is lightweight and can easily be shipped or moved.

## 3.6 Inverse Requirements

State any \*useful\* inverse requirements.

## 3.7 Design Constraints

The design of the HDS will remain quite simple. The raspberry pi is a small device, so it wont take up much room, so it can be located almost anywhere in the house. The raspberry pi also has a plasitc case around the microchips to protect them. The hardware only requires a power outlet in order for the device to be turned on. Also, the raspberry pi will need to have internet access in order for the device to send data to the cloud, so internet is required. The pi also uses an SD card which has 16 GB of space available.

## 3.8 Logical Database Requirements

The database being used is all contained in the cloud. The device will be using Microsoft Azure to send data to the cloud. The cloud will store up to a weeks worth of video for the user to look back on. The raspberry pi will take a picture of the the room every 2 seconds, and send those pictures to the cloud. In order to be more space efficient, the pi will use pictures, instead of recording video, due to the high amount of data video capturing requires.

## 3.9 Other Requirements

The user will need a cell phone in order to access all of the data captured by the device. The pi will use senors to send specific data, which can then be read on a mobil device. If the user doesn’t have a mobile phone then they won’t be able to fetch the data.

# 4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS’s requirements.

## 4.1 Sequence Diagrams

## 4.3 Data Flow Diagrams (DFD)

## 4.2 State-Transition Diagrams (STD)

# 5. Change Management Process

Any and all changes to the SRS document will go through the interim project master (scrum master) for tweaking and approval. The changes will be suggested briefly during the five minute discussion before class time. The tweaking will be done by consulting all members of the team for their input, but the final decision lies with the scrum master. Once approved, the scrum master will assign the responbility to the appropriate team member(s).

# A. Appendices

*N/A*

## A.1 Appendix 1

## A.2 Appendix 2

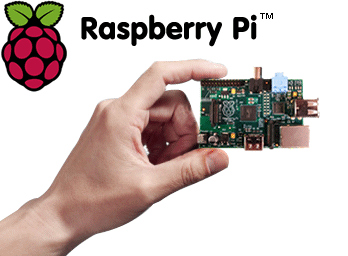
Tianqi Huang O’Dell

Coury Ferson

Michael Mcbride

Tianqi Huang

Christian Kirk



**Contents**

**Section 1 - Project Description**

*1.1 Project*

*1.2 Description*

**Section 2 - Overview**

*2.1 Internal Software Data Structure*

*2.2 Global Data Structure*

*2.3 Temp Data Structure*

*2.4 Database Description*

**Section 3 - System Architecture**

*3.1 System Structure*

*3.2 Description for Pi*

*3.3 Description for Azure Cloud*

*3.4 Dynamic Behavior for Home Defense System*

**Section 4 – User Interface Design**

*4.1 Description of the User Interface*

*4.2 Interface Design Rules*

*4.3 Components Available*

**Section 5 – Restrictions, Limitations, and Constraints**

**Section 6 – Testing Issues**

*6.1 Classes of Tests*

*6.2 Expected software Response*

*6.3 Performance Bounds*

*6.4 Identification of Critical Components*

**Section 1 – Project Description**

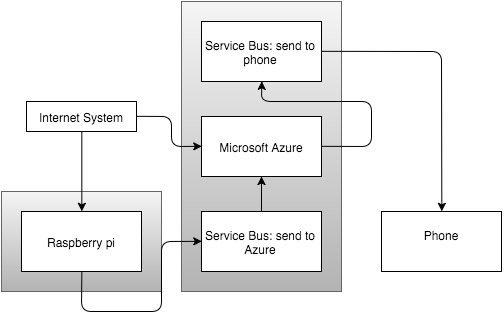
* 1. **The Home Defense System**

**1.2**

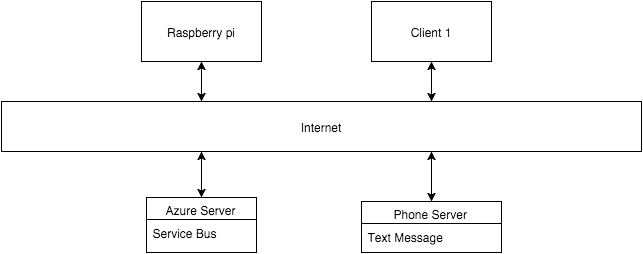
The Home Defense System is an alarm system using the Raspberry Pi and the cloud. By using this system and triggering special alarms for the user we are able to detect motion, take pictures of the motion detection and send it currently to Drop Box, and also we are able to retake the temperature every minute to make sure there is not a fire or a pipe burst. By having a system that is a fair price and easy to use we are able to give the middle class consumer some home security without having to break the bank. Our consumer will be able to protect their home and their loved ones by simply taking a brief moment to set up the Pi.

**Software Design**

We have a few different systems we will be using in our project. We use an internet service, the raspberry pi, and Microsoft Azure. All of the systems must communicate with each other in their own way. The pi must talk to the cloud, and the pi cloud must talk to the users phone. In order for the systems to be able to send and receive all of their data they must have internet or the fetch will fail.



Our system will be following the Client-Server architecture. We rely on our services to communicate with each other via the internet so it fits our needs the best.



2. Data design

2.1 Internal software data structure

The Home Defense System internal data structure is divided into two parts: The raspberry pi temperature trigger and raspberry pi motion detection trigger.

1. In the raspberry pi local memory, motion detection trigger will set as Boolean type data. It will send true when the camera detection detect a motion happened. After cloud receive trigger, the cloud will decide to send the break-in alarm to the user or not.
2. In the raspberry pi local memory, temperature detection trigger will set as Boolean type data. It will send true when the temperature detection detect the temperature is higher than user set value. After cloud receive trigger, the cloud will decide to send the fire alarm to the user or not.

2.2 Global data structure

Once the detection detect a motion or aberrant temperature, it will create a data file which contain the motion frame or temperature log. Then this file will send to cloud and finally send to user device.

2.3 Temporary data structure

The temperature and frame capture by detection each second are temporary data structure which the data will disappear after the algorithm analyze them.

For the snap of frame which detection think it is motion. The home defense system will create a frame file at local pi memory, then it will send to cloud and destroy at local pi memory.

2.4 Database description

In this project, we don’t use database.

3. Architectural and component-level design

3.1 System structure

The Home Defense System is broken up into two major components: a raspberry pi that data collection and analyze device, the cloud that policy engine and alarm sender system.

The cloud is a system that can receive pi’s trigger and put the trigger to policy engine. Then, decide send the alarm to the user device or not.

3.1.1 Architecture diagram

Home Defense System

Raspberry Pi

Azure Cloud

User Device

3.2 Description for Raspberry Pi

3.2.1 Processing narrative (PSPEC) for Raspberry Pi

The raspberry pi is a device that include two major detection, motion detection and temperature detection. Both detection will collection data and analyze data. After analyze, if there are motion indeed happened it will send the motion detection trigger to cloud. And if temperature aberrant, it will send the temperature trigger to cloud.

* + 1. Raspberry Pi interface description

The input interface of pi will divided by two parts. The one is the video capture by pi’s camera and another is the temperature collect by pi’s temperature detection.

The output interface of pi is the trigger of motion detect and temperature detect. Furthermore, if motion detect find motion happened, the snap frame of motion detection will send to cloud.

* + 1. Raspberry Pi processing detail

Firstly, Raspberry Pi should running under the cv virtual environment. Secondly, the Pi should connection the cloud. Thirdly, the motion detection python program and temperature detection c program should run. Finally, after everything run correctly, the Pi should send temperature trigger per second and motion trigger when motion was detected to cloud.

* + - 1. Design Class hierarchy for Raspberry Pi

Raspberry Pi

Motion Detection

Temperature Detection

Get Video

Send trigger

Get Tem.

Send trigger

Analyze Video

Analyze Tem.

s

s

s

* + - 1. Restrictions/limitations for Raspberry Pi

1. The Raspberry Pi only work under internet connected.
2. The Raspberry Pi need professional set up before user use.
   * + 1. Performance issues for Raspberry Pi

The Raspberry Pi will send trigger to cloud per second and motion trigger will only send to cloud when motion detected. If motion detected by detection, the FPS on live screen will lower 20 and frame will have delay.

* + - 1. Design constraints for Raspberry Pi

1. The Raspberry Pi’s camera is not night vision, thus the Pi only can install light controlled place.
2. The Raspberry Pi’s CPU is not good enough, thus the Pi will slower execution when large motion detected.
3. Slow WiFi network card may cause trigger send lag. Internet is suggested.
4. The Temperature Detection sometime has bad data collection. But still reliable at most time.
   * + 1. Processing detail for Motion Detection of Raspberry Pi

The Motion Detection is the core function of the Home Defense System.

3.2.3.5.1 Processing narrative for Motion Detection

The detection will get video by Raspberry Pi camera, and analyze each frame then detect weather the motion happened. Once motion detected is positive, then send trigger to the cloud with the snap picture file will created at Pi local memory, and send immediately to the cloud. After file sent, the local temporary file will deleted.

* + - * 1. Algorithmic model for Motion Detection

The background subtraction is the main algorithm of motion detection. The video will capture by Raspberry Pi camera, the pi\_surveillance.py python program will check each frame which capture by camera, program take the weighted mean of previous frames along with the current frame. Thus the background will dynamically adjust by lighting condition and static motion (like put a cup in front camera, the cup will detect as motion, but after cup stand there a while, the cup is not count motion anymore).

After motion detected by motion detection, a Boolean value will send as trigger to the cloud policy engine.

* + - 1. Processing detail for Temperature Detection of Raspberry Pi

The Temperature Detection can help user avoid fire hazard.

* + - * 1. Processing narrative for Temperature Detection

The Temperature Detection is built by DHT11 Humiture Sensor. This sensor can collection temperature and humidness per second. During the detection running, a temp\_log.txt file also write as temperature log for reference and back up. If the temperature higher than 150 degree in Fahrenheit, the temperature trigger will send to cloud.

* + - * 1. Algorithmic model for Temperature Detection

The Temperature Detection program write by c program. It provided by DHT11 sensor provider. The temperature and humidity will show on screen per second. Once the temperature higher than 150 degree, a Boolean value will send as trigger to cloud policy engine.

3.3 Description for Azure Cloud

3.3.1 Processing narrative (PSPEC) for Azure Cloud

Microsoft Azure is online software used to create various applications. We are choosing to use Azure for its Service Bus application. The Service Bus is used to fetch information from the pi and send a message to the user.

* + 1. Azure Cloud interface description

The interface is very user-friendly. Azure comes with a fully loaded dashboard that displays all of the users services, allowing them to easily select, modify, or delete anything they choose

* + 1. Azure Cloud processing detail
       1. Design Class hierarchy for Azure Cloud
       2. Restrictions/limitations for Azure Cloud

In order for the cloud to successfully get the information from the pi it must be connected to an Internet source. Also, the cloud costs money every time you fetch data. So the more data being fetched, the more money will be required to keep the service active.

* + - 1. Performance issues for Azure Cloud

Azure is very efficient. Although the fetch requests don’t get to Azure immediately, there is only a 5-10 minute delay, which is not of much concern.

* + - 1. Design constraints for Azure Cloud
      2. Processing detail for Operation 1 of Azure Cloud
         1. Processing narrative for Operation 1

When the Service Bus is created it gives the user a key for each operation. In order to successfully send data to the cloud the user must use those keys in a desired program to link to the cloud. Our system uses a python program to send a message to that key every time our test check is met.

* + - * 1. Algorithmic model for Operation 1

The Algorithm used for the sending operation is very simple. It consists of a for – loop that sends the current data to the cloud every time a check is failed. It doesn’t take much processing for the program to run these checks.

* + - 1. Processing detail for Operation 2 of Azure Cloud
         1. Processing narrative for Operation 2

After the cloud receives a message from the pi it will then be required to notify the user of the information gathered. In our case the cloud will send a text message to the user with a send operation.

* + - * 1. Algorithmic model for Operation 2

Much like the first operation, the algorithm needed to send the user a text message is very simple. It uses a while loop to and once the loop becomes true, it will send a text message to the user. The user however must have a valid phone number or the message cannot be sent.

* 1. Dynamic Behavior for Home Defense System

The Raspberry Pi interact Azure Cloud by trigger, and Azure Cloud interact Pi by service bus. And the Azure Cloud will send alarm to user device by internet.

3.4.1 Interaction Diagrams

Raspberry Pi

Microsoft’s Azure cloud

User Device

[ Motion\_Detection\_Trigger == True ]

Motion\_Alarm\_on()

Data\_analyse()

allt

Alarm\_reveived\_confirmation

Send\_snap(snap\_frame)

[ Temp\_Detection\_Trigger == True ]

allt

Temp\_Alarm\_on()

Alarm\_reveived\_confirmation

Send\_Trigger ()

Loop

4.0 User interface design

The user will be able to choose from a range of temperature values so that they can specifically set a value that when exceeded, the user will receive an SMS message. The user can also set a certain time for the Pi to look for motion detection.

(i.e. If temp > 90 send SMS)

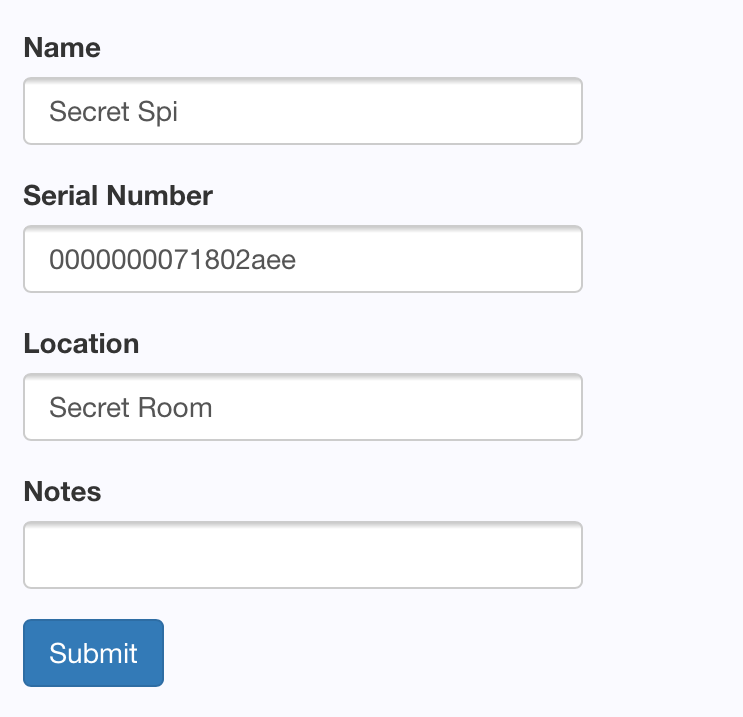
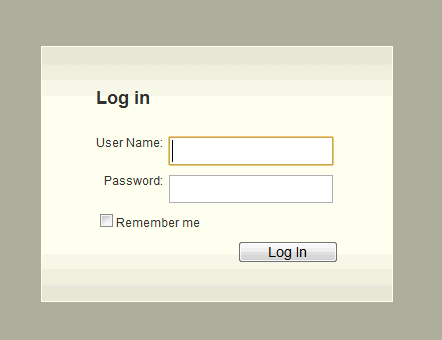
(i.e. RecordTime = 11:00 a.m. – 1:00p.m.)

4.1 Description of the user interface

The user will be able to log on to the Cloud service and set values for temperature and time.

4.1.1 Screen images

Standard Log on GUI. We also provide an interface to register your Raspberry Pi.



4.1.2 Objects and actions

The user will be required to sign up for the Home Defense System. After signup they will be taken to a page where they can register their Pi, and from there they can modify the settings for when the Pi will record video and when the Pi will record temperature. They can also modify the criterion that needs to be met in order to receive an SMS alert.

4.2 Interface design rules

The user will *not* be able to enter non-sensible data for the temperature and motion detection values. (i.e. Temperature = 2000) (Time = 1 p.m. – 12 p.m.)

4.3 components available

The main GUI component will be hosted through Microsoft Azure.

4.4 UIDS description

*N/A*

5.0 Restrictions, limitations, and constraints

Some hardware limitations that we have to work around is the limited processing power of the Raspberry Pi itself, along with the limited memory. While the memory is more than sufficient enough for our programs to run on, the picture data generated will have to be continuously deleted to make room for new data to be stored.

We are constrained by using Microsoft Azure for the cloud services, which means that the Raspberry Pi also requires a constant internet connection.

6.0 Testing Issues

We will mainly focus on testing the policy engine and the ability for the user to define policies. There will be preliminary tests about data transfer from the Raspberry Pi to the cloud and the cloud to the user, data storage, and data deletion after a specified time.

6.1 Classes of tests

In terms of black-box testing, we will test the ability of the system as a whole to transfer data and alert the user when necessary. The inputs will be either motion or temperature detected and then alerting the user. White box testing will be done, mainly with making sure that our system can handle any errors that bad input from the camera or temperature sensor would give. Also, making sure that our logic checking is correct for the user defined policies.

6.2 Expected software response

We expect the software to correctly alert the user if motion or temperature triggers are sent during the times that the user wishes to be alerted, as per their policies. Otherwise, the system should not falsely alert the user. If the user tries to enter a logically invalid policy, the system should inform them and allow them to change the policy.

6.3 Performance bounds

After motion is detected or temperature is detected outside of its acceptable values and the policy conditions are met, the user should be notified within 10 seconds.

6.4 Identification of critical components

The input devices on the actual Pi itself will need testing and checking to ensure that erroneous values are properly handled. The only other critical part is checking the user defined policies to validate them to avoid false alerts being sent.

**Task 3**

We chose the MIT license because we wanted our software to be open source. Open source gives the opportunity for programmers to make better programs and that’s exactly what our team wants to see. If someone can make our product better, we’re all for it. As a team we want to share what we’ve learned with the programming community, that’s why this license works best for us.

**----------------------------------------------------------**

**The MIT License (MIT)**

The MIT License (MIT)

Copyright (c) <year> <HomeDefenseSystem >

Permission is hereby granted, free of charge, to any person obtaining a copy  
of this software and associated documentation files (the "Software"), to deal  
in the Software without restriction, including without limitation the rights  
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
copies of the Software, and to permit persons to whom the Software is  
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in  
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,  
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE SOFTWARE.

**Home Defense System Test Plan**

**Team 5**

**Coury Ferson, Tianqi Huang, Christian Kirk, Michael Mcbride, Ian O'Dell**

**# = Non-Functional**

* 1. – Motion Detection
  2. – **Test Motion Power**

Precondition: Make sure the camera is plugged into the pi.

How to test: Run function to check for power.

Post-condition: Determine whether the sensor has power.

* 1. – **Test Motion Detection**

Precondition: The camera is on and the specific motion detection program is running.

How to test: Check if the Boolean statement is true or false.

Post-condition: Motion will or will not be detected

* 1. – Temperature
  2. – **Test Temperature Power**

Precondition: Make sure the temperature sensor is plugged into the pi.

How to test: Run function to check for power.

Post-condition: Determine whether the sensor has power.

* 1. **- Test Data Gathering**

Precondition: The get() temperature function is running.

How to test: Run the program to see if the function collects the proper data.

Post-condition: Display temperature.

* 1. **Check Temperature Range**

Precondition: A temperature range is entered.

How to test: Print out the temperature that is exceeding or below the temperature range.

Post-condition: The range valid or invalid.

3.1 – Cloud

**3.2** – **Check Authentication**

Precondition: Credentials are entered.

How to test: When the program is ran, authentication fails or succeeds.

Post-condition: The data is sent to the cloud.

**3.3 – Test SMS**

Precondition: Policy criteria met.

How to test: Give inputs that meet the criteria.

Post-condition: The SMS message is sent.

* 1. – Performance/System

**4.2** – **Test System Power #**

Precondition: Make sure the system is plugged into a power source.

How to test: Turn on the system to check for power.

Post-condition: The system will turn on or stay off.

* 1. **– Test Internet Connection #**

Precondition: Have an internet source.

How to test: Try to send data (i.e. temperature, picture) to the cloud.

Post-condition: The data is received or not received.

* 1. – **Test SMS Response Time #**

Precondition: All components set up and functional.

How to test: Invoke trigger and monitor time taken to receive an alert.

Post-condition: Time taken to receive the message is recorded.