Pi-Surveillance

Final Year Project Report

DT211C
BSc in Computer Science (Infrastructure)

Dean Ryan
Aneel Rahim

School of Computing
Dublin Institute of Technology

Abstract

In the last number of years, there has become a growing concern on burglary and related offences throughout Ireland. This has influenced the development of smart technologies such as smart lock systems, smart alarm systems and more importantly, smart monitoring systems. Unfortunately, smart security technologies can be very expensive and for this reason may appear less of a need for the consumer.

The aim of this project is to provide a cost effective high end monitoring system for home owners. Pi-Surveillance is an application for both web and android devices.
Users can login to view a live camera feed attached to a raspberry pi server whether they are at home or away. The camera runs on motion detection software attached to a pan tilt zoom bracket that tracks movement detected within an image.

When motion is detected, an email notification is sent to the user. Videos are pre captured and saved to an external drive located on the server. The media is also accessible using the web and android application which provides an instant easy view of the saved captured videos.

The web application provides admin features so that home owners can add and allow access to friends and family members. He/she may also change and add cameras as they please. This is necessary to keep an efficient, safe system.

This system incorporates a various amount of architectures which includes Java and XML for the android application, PHP, CSS and JavaScript for the web application and finally MySQL to store data.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

____

Dean Ryan

Acknowledgements

I would like to thank my supervisor, Aneel Rahim, for his valuable and constructive suggestions during the development of this project.

To the head supervisor Damian Bourke, I would like to thank him for his understanding and cooperation of personal circumstances faced during my final term.

Many thanks to Brian Keegan for his support throughout college and particularly providing me with the resources needed to complete my project.

Finally, I wish to thank my family for their dedication and the many years of support during my undergraduate studies that provided the foundation for this work

Contents

Figures

Chapter 1: Introduction

Introduction

Security system at an affordable price are very hard to come by nowadays. With security companies adding costs such as support and repairs to the bill, it can all seem a bit much. However smart technologies like wireless monitoring systems and smart locks have risen and are becoming much more familiar.

Project Aim

The aim of this project is to develop an application for both web and android devices that will provide a live stream of video surveillance from a camera attached to a raspberry pi server. The system will help aid in the prevention of burglary and theft related incidents.

Using the web and android application, users can not only view live feed from a tracking camera on the server but also gain instant access to saved media that is captured by the motion software. The system sends email notifications to users, allowing them to view the safety of their home from any location provided they have internet access.
The system is cost effective, Intuitive, and consists of multiple features that can be found in higher over paid systems like Motorola, BT and Belkin.
Figure 1 below is a brief overview of the system architecture:

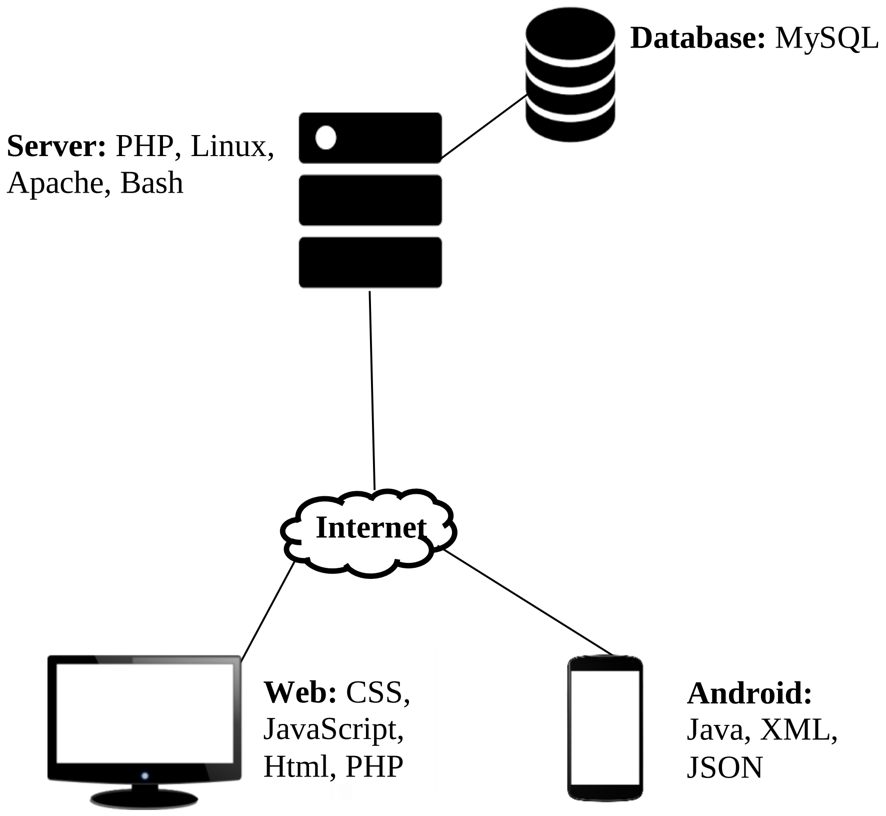

Database: MySQL
Server: PHP, Linux, Apache, Bash
Internet
Web: CSS, JavaScript, Html, PHP
Android: Java, XML, JSON

Figure 1: System Architecture

Structure of Document

Chapter 2: Research

This chapter reflects on the information gathered before and during the development process of this system. The chapter will give a brief overview on the history of surveillance systems, why they are so important and also why they have become so costly. A detailed overview of various different monitoring systems will be provided comparing them to the system at hand. The systems will then be evaluated using Nielsens Heuristics.

Chapter 3: Requirements Analysis

Chapter 3 will discuss information relating the functional and non-functional requirements of the system. The requirements will then be divided into tables to display an outline of requirements of the system.

Chapter 4: Technologies Researched

This chapter examines the technologies that may and have been beneficial to the system. It will discuss the technologies in detail and evaluate a solution as to which technologies best suited the system and why.

Chapter 5: Design

Chapter 5 intends to give a clear overview of the system using various diagrams and tables. A full technical architecture of the system will be presented to further explain the technologies used and how they are used. A number of different methodologies will be reviewed while explaining the most appropriate method suited for the system.

Chapter 6: Implementation

In this chapter, a detailed overview of the technologies used are shown and explained. The chapter will show how the system was implemented and will provide some code examples of the functionality working. The algorithm used for the motion detection will also be included here.

Chapter 7: Testing

Following chapter 6, an examination on the components used on the system will be elucidated. This will include a number of test cases representing the android, web and server section of the system. They intend to give a clear outline on the results that the system produces.

Chapter 8: Results and Evaluation

This chapter will give a reflection on the overall project. It will take a look at what has been achieved and what has changed since the beginning of the project. It will also discuss features that have been dropped and added to the system.

Chapter 9: Conclusion

Chapter 9 will then expresses the results and evaluation of the system in general terms including possible future plans for the system.

Chapter 2: Research

Introduction

The following section provides brief overview of the history and development of surveillance systems overtime. It examines the information gathered from technologies and similar systems in an effort to achieve a more feasible project. The information obtained will be divided into subsections focusing specifically on the researched systems architecture and their relevant features. Nielsen's heuristics for interaction design will be practiced in order to correlate the associated systems with the apposed system [1].

CCTV

In 1933 the world's first security camera was invented by an amateur photographer who went by the name Mr. Norbury. Whilst his invention was largely ignored around the time of creation, he has since been hailed as the father of camera surveillance. Using an ordinary box camera placed inside a chicken hut, he was able to capture pictures of the man who was stealing eggs for weeks prior to the capture. The photos were later used in court and justice was served [2]. It wasn't until the late 1940's that the systems were used on a commercial bases. Systems consisted of cameras and monitors that could only be used for live monitoring.

Figure 2: Box Camera

Eventually reel to reel systems were developed and recording was made possible. It wasn't until the emergence of the video cassette recordings (VCR) where surveillance systems became widely used. Users could review recorded footage whenever needed. Systems involved the use of analogue cameras which needed coax cables that had to be directly connect to the VCR.

Since then, the use of video surveillance, more commonly known as CCTV (closed-circuit television), has become very popular. We have seen solutions like multiplexing where syncing and recording between multiple cameras on one screen has been made possible. Digital video recordings (DVRs) was also a huge step for CCTV as it was more user friendly and meant no more hassle of leads or video tapes.
Today, top companies like Hikvision use network video recording (NVR) systems. These systems work by encoding and processing video in cameras and then streaming the footage to NVRs for storage or remote viewing [3]. NVR systems can now include internet protocol cameras (IP). IP cameras can send and receive information over a network and internet connection. The problem with such systems is that the equipment can be very expensive to purchase and maintain.

However, IP cameras have now been found useful with other technologies such as web and mobile phone applications. Many different applications are now available to download which provide live stream from IP cameras over networks.

Security Costs

Security systems generally consist of two main factors that are purchase price, and ongoing monitoring. Although both services can differ in price, users might find that the final decision could depend on the comfort of when he/she may be away from home. There are several types of security system that users can choose to protect their home. The consumer may wish to purchase a monitored or unmonitored system. Monitoring systems are often accessible any time of day but tend to be more expensive. This is because monitoring system require 24/7 protection whereas unmonitored don't. Unmonitored systems are generally systems designed by the consumer for their own benefit. This may work out as more affordable solution but some may argue that it can be less technical and therefore become more vulnerable. Companies like Motorola, Netgear and Logi have designed systems where users can view live video feed from any of their purchased cameras. However the average camera provided by each company costs roughly 100 to 130 euro.

Similar Existing Solutions

The following research consists of similar systems identifying the characteristics and features which lead to the success of each system. This was carried out in an effort to develop the best possible overall system. A further analysis will then be provided on each systems functional requirements which will then be compared against Nielson's Heuristics in order to rule out weaknesses which may be found in the interface design. The following systems are highly rated developments which have been found on the android app store throughout the research of this report.

Raspberry Pi Camera Viewer

The application allows users to connect and stream video footage from their raspberry pi device [Figure 3]. Raspberry Pi Camera Viewer uses the Gstreamer framework which works on multiple OS's and supports a broad coverage of multimedia technologies. The application offers users to build their own pipeline on their raspberry pi which will still work with the application.

Despite the attractive layout and simple UI provided by Raspberry Pi Camera Viewer, it does contain several flaws which users may find bleak. Flaws include:

- In order to run the system, the user needs to install the Gstreamer framework on their raspberry pi.
- The user also needs the set the output of the stream on the raspberry pi in order for it to launch
- Full screen mode is unavailable
- Video cannot be recorded
- The application only works over local network
- Does not support image filtering or zoom features

The application does however stand out in other areas:

- The application provides users with the option to add multiple cameras

- Users can scroll between multiple cameras connected to the network
- The option to hide controls is available which represents a nice GUI
- Play and Pause is also available
- Users can setup widgets on the home screen for quick access to the cameras

As previously stated, the application provides a unique layout and a very attractive UI. A feature that has been considered throughout the development process of the systems application.

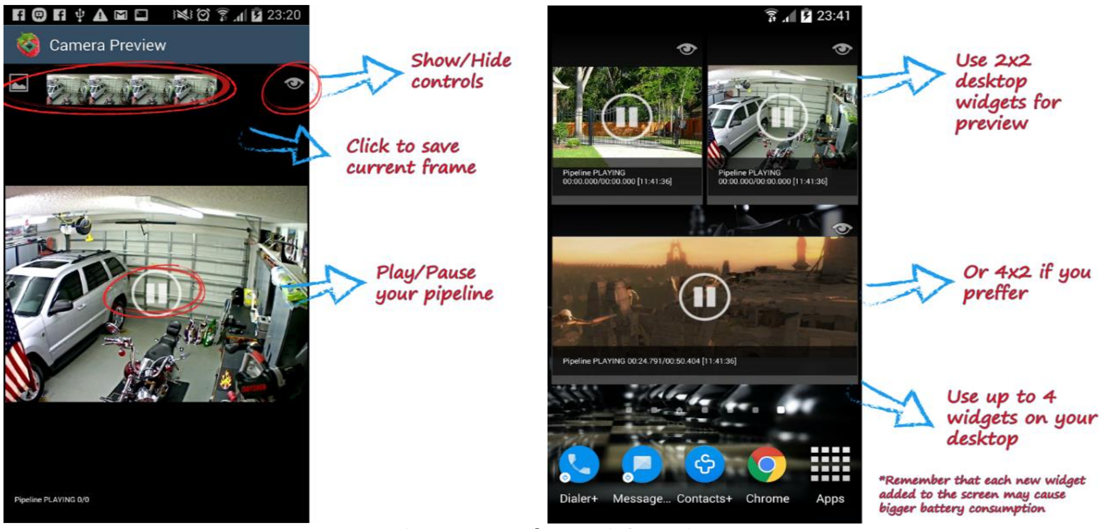

Camera Preview
Show/Hide controls
Click to save current frame
Play/Pause your pipeline
Pipeline PLAYING 0/0
Use 2x2 desktop widgets for preview
Or 4x2 if you preffer
Use up to 4 widgets on your desktop
*Remember that each new widget added to the screen may cause bigger battery consumption
Dialer+
Message...
Contacts+
Chrome
Apps

Figure 3: Raspberry Pi Cam Viewer

p2pCamViewer

P2pCamViewer is a free android application that provides live streaming for IP cameras over LAN and WAN connections [Figure 2.3]. Users can login and save cameras connected a network that is provided. The application includes a network scan feature which displays a list of available IP and ports on the network.

Although the application provides a number of quality features, it lacks in other areas such as the layout and design of the application which is rather plain looking. Also, the application provides users with register and login accounts. However, it does not provide users with the ability to edit or delete their accounts. This shows a poor development flaw within the application. The application does not support filtering or home screen access to camera views like provided by Raspberry Pi Camera Viewer.

Unlike Raspberry Pi Camera Viewer, the application allows the users to record video on multiple cameras. It includes an abundant of other features which include:

- PTZ options are available on each camera. This offers users with supported cameras the ability to zoom in and out of specified locations within the camera view.
- An effective quality of the application is its capability to access the IP camera over local and wide networks. This offers a great security feature to the application

- P2pCamViewer includes a listen and talk feature which can be very beneficial. For example is users are not home this would be of great use.
- The most effective quality in my opinion is the use of network scanning. Users can scan local networks for available IPs. This rules out complications and offers a more simplistic approach when needing to connect to cameras over a network

As seen above, although the application does not provide an attractive layout and interface, it does contain a great value of quality features that in fact have been used in the development process of this system. After researching a number of different applications, it had been concluded that not many applications offer users access over both local and public networks.

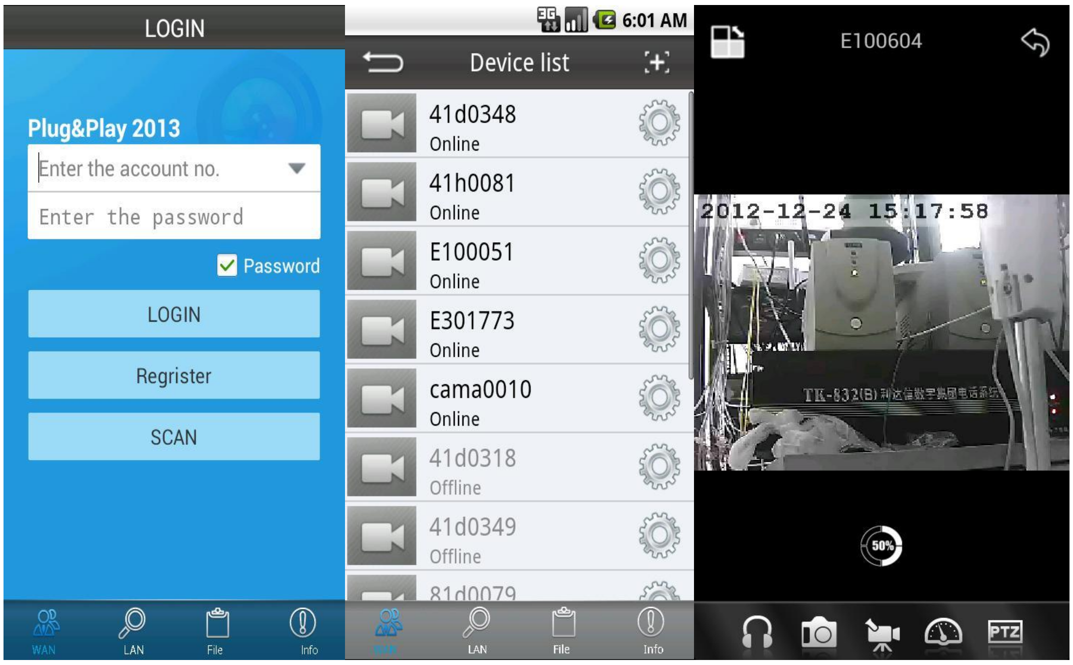


Figure 4: p2pCamViewer Layout

Video Surveillance Ivideon

Ivideon is a cloud based video surveillance application available on both android and IOS devices [Figure 2.4]. It provides users with access and reliability without complexity. Users can record and view recordings made through the device. The applications supports the use of IP cameras over networks and provides the highest end of features an application can provide.

Ivideon is a well-developed application that provides a near exact system to Pi-Surveillance. However, there are some features that have been disregarded Pi-Surveillance. The application provides the ability to record and play back video via Ivideon cloud server. Pi-Surveillance supports users with the use of external device storage such as Hard-drives, memory sticks etc. This guarantees a more secure approach to video storage and limits access to vulnerable threats within the cloud. The application also costs money to download. Yet another unpopular feature for the application as many users prefer free applications to priced applications.

The application has proven to be the most similar to Pi-Surveillance. It provides almost every feature implemented with Pi-Surveillance. Some features include:

- Record and store videos over the cloud.
- Search and view recorded video surveillance. Users can playback video stored on the cloud.
- Just like P2pCamViewer, it provides users with the PTZ feature.
- Receive notifications via email of suspicious movements or sounds.
- Share camera links over social networks

The application provides a lot of features which have been implement with Pi-Surveillance. A feature found very beneficial is the use of notifications upon detected motion.

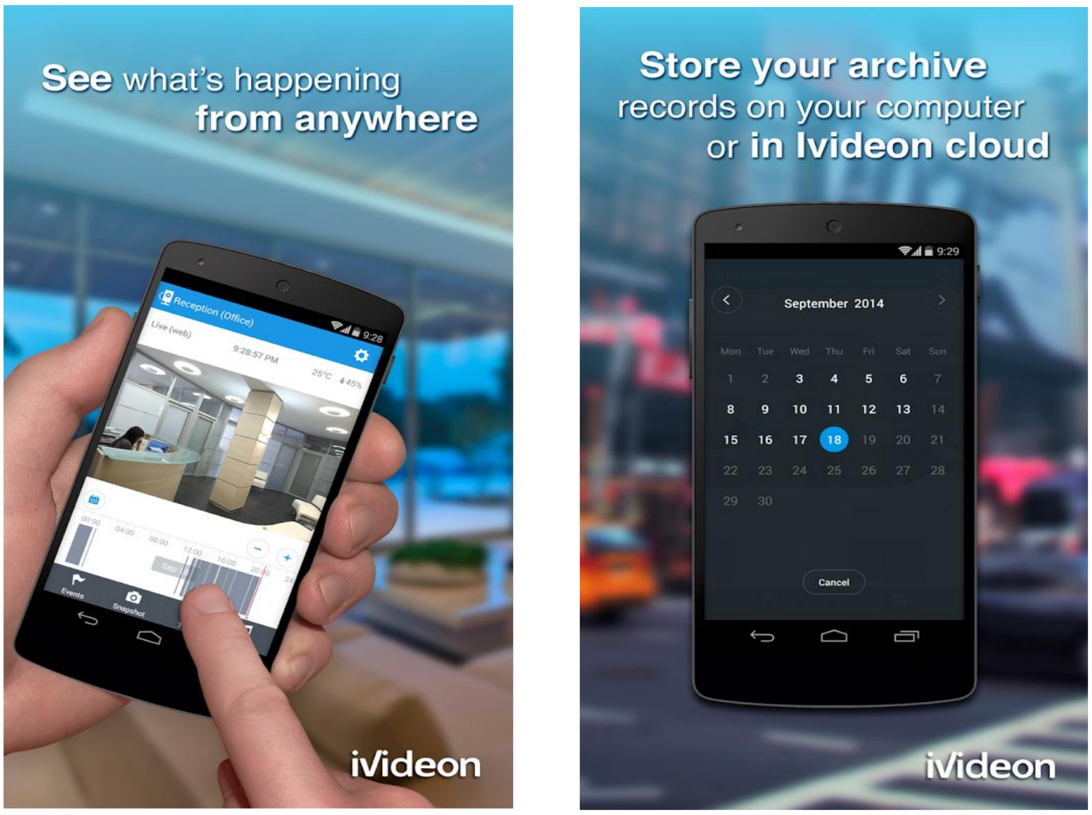

See what's happening from anywhere
iVideon
Store your archive records on your computer or in Ivideon cloud
September 2014
Cancel
iVideon

Figure 5: Video Surveillance Lvideon

Nielsens Heuristics

A heuristic evaluation is a usability inspection method for computer software that helps to identify usability problems in the user interface (UI) design [5]. The evaluation consists of 10 principles which are each used to evaluate system features such as usability, efficiency and effectiveness.

1. **Visibility of system status**
   The system should always keep users informed about what is going on, through appropriate feedback within reasonable time [4].

2 **Match between system and the real world**
The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order [4].

3 **User control and freedom**
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo [4].

4 **Consistency and standards**
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions [4].

5 **Error prevention**
Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action [4].

6 **Recognition rather than recall**
Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate [4].

7 **Flexibility and efficiency of use**
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions [4].

8 **Aesthetic and minimalist design**
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility [4].

9 **Help users recognize, diagnose, and recover from errors**
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution [4].

10 **Help and documentation**
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large [4].

Comparison of Heuristics

Using the process of Nielsens Heuristics as shown above, the information gathered can now be evaluated. In order to proceed with this process, individual tables have been created outlining each step for each system.

Visibility of Systems Status	✓
Match Between System and the Real World	x
User Control and Freedom	x
Consistency and Standards	✓
Error Prevention	✓
Recognition Rather Than Recall	✓
Flexibility and Efficiency of use	x
Aesthetic and Minimalist Design	✓
Help Users Recognise, Diagnose and Recover from errors	x
Help and Documentation	x

Figure 6: Raspberry Pi Viewer

Visibility of Systems Status	x
Match Between System and the Real World	x
User Control and Freedom	x
Consistency and Standards	✓
Error Prevention	x
Recognition Rather Than Recall	✓
Flexibility and Efficiency of use	✓
Aesthetic and Minimalist Design	✓
Help Users Recognise, Diagnose and Recover from errors	x
Help and Documentation	x

Figure 7: p2pCamViewer

Figure 8: Video Surveillance

Visibility of Systems Status	✓
Match Between System and the Real World	✓
User Control and Freedom	✓
Consistency and Standards	✓
Error Prevention	x
Recognition Rather Than Recall	✓
Flexibility and Efficiency of use	✓
Aesthetic and Minimalist Design	✓
Help Users Recognise, Diagnose and Recover from errors	x
Help and Documentation	x

Chapter 3: Requirements Analysis

Introduction

The requirement analysis sections consists of information relating to the functional and non-functional requirements of the system. The requirements help give a clear outline on what the system exactly does. After surveying 5 home owners throughout a local housing estate, the results gathered were used to create the tables below

.

Hardware & Software Requirements

- Raspberry Pi Model 2B
- Raspbian Jessie OS
- Pi Camera
- Android Device
- Android Studio
- Laptop (Windows 10 home) using Putty for SSH
- BASH
- MySQL
- HTML
- PHP
- Java

Functional Requirements

No.	Type	Description	User	Priority
Func-1	Android/Web	Users can login	Users/ Admin	High
Func-2	Android/Web	Users are a logged in user	Users/ Admin	Medium
Func-3	Android/Web	Users can log out	Users/ Admin	Medium
Func-4	Web	Add users	Admin	Medium
Func-5	Web	Edit user credentials	Admin	Low
Func-6	Web	Add camera	Admin	High
Func-7	Web	Update camera	Admin	High

Func-8	Web	Add media	Admin	High
Func-9	Web	Update media	Admin	High
Func-10	Web	Delete saved media	Admins	Low
Func-11	Android/Web	View saved media	Users/Admin	Medium
Func-12	Android/Web	View video stream	Users/Admin	High

Figure 9: Functioal Requirements

Non-Functional Requirements

No.	Type	Description	Priority
Func-1	Performance	The android and web application will provide fast reliable stream at suitable frames per second	High
Func-2	Availability	The camera view will be available from anywhere	High
Func-3	Reliability	The system will be constantly refreshing to provide a stream of images	Medium
Func-4	Compatibility	Video streaming will provide multiple resolutions for clearer images	Low

Figure 10: Non-Functional Requirement

Chapter 4: Technologies Researched

Introduction

In order to determine an effective system, a full research and evaluation of technologies were required. The following chapter consists of a list of well documented technologies that were used in the development of Pi-Surveillance.

Some of the technologies researched are compared with others and have therefore led to unused technologies.

Android

There are several reasons one might develop his/her application using android. The main reason for a Pi-Surveillance android application is due to testing purposes. Having an android phone means that one will not have to use the android emulator and therefore have quicker results. Also android is cheaper to develop for and runs on a simple platform called android studio which is later discussed. Technically speaking, android is easily ported and is supported on multiple operating systems.

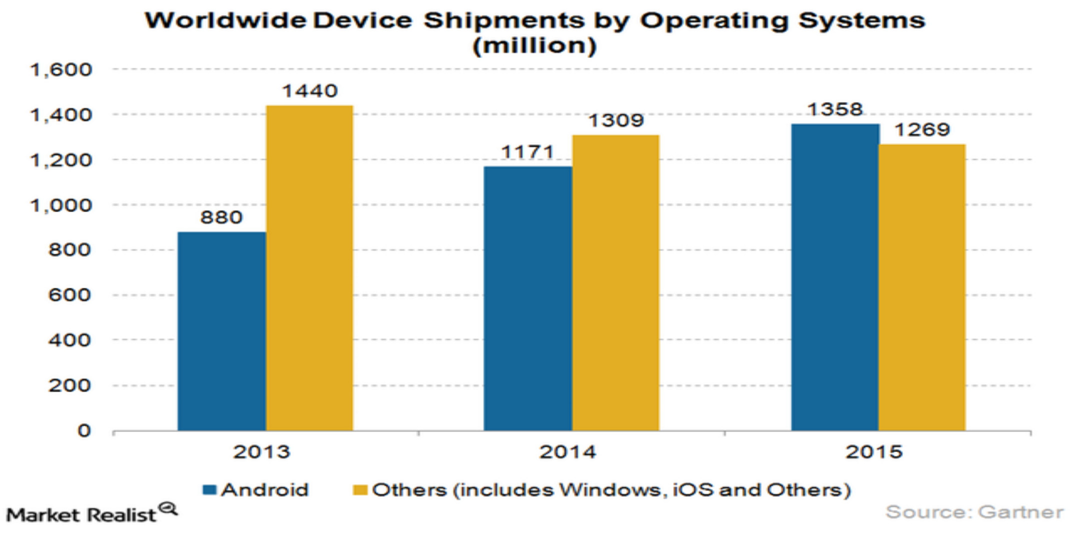


Figure 11: Yearly Shipment Orders

This immediately gives an advantage over window and apple devices. In order to develop for iOS, users would need to use a MAC. Also registration on the Apple App Store is more expensive in comparison to android.

Raspberry Pi 2 Model B

A Raspberry Pi 2 has been used as the server for Pi-Surveillance. It was originally planned that the use of an Arduino model would be more appropriate because of previous experience. However raspberry pis consisted of a better spec at the same value. The Pis spec range included; HDMI output for testing, USB ports for external storage (media files) and Ethernet port for networking.

The Raspberry Pi provides support for multiple operating systems which include Raspbian, Ubuntu Mate and Fedora to name a few. Subsequently it's believed that the most popular OS used is Raspbian.

	Arduino Uno	Raspberry Pi 2 Model B
Price	30	35
Size	7.6 x 1.9 x 6.4 cm	8.6 x 5.4 x 1.7cm
Memory	0.002MB	512MB
Clock Speed	16Mhz	700Mhz
On Board Network	None	10/100 wired Ethernet RJ45
Multitasking	No	Yes
Input Voltage	7 to 12 V	5 V
Flash	32KB	SD Card
USB	One, input only	4 peripherals
Operating System	None	Linux Distributions
Integrated Development Environment	Arduino	Linux support, IDLE, Scratch
HDMI	No	Yes
DSI	0	1

Figure 12: Arduino Vs Raspberry Pi 2B specs

Java for Android

Java has been used to develop the android application for the system. Mainly due to previous experience developing with java. Androids Network Service Discovery (NSD) allows users to search and select devices over local networks. This idea was dropped from the system as there was not enough time to finish it. Note that for the purpose of this project users will not have access to features provided by the raspberry pi server. Therefore the application will be used mainly as a view for user IP cams or self-developed similar systems.

XML for Android

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML [7]. It plays a huge part in the exchange of wide varied data and helps structure it for interfaces such as web and android devices.

PHP

Php is a server scripting language that helps build dynamic, interactive web pages. It helps to evaluate form data sent between browsers, talk to a database such as MySQL and even send and receive cookies [8]. Pi-Surveillance is mostly built using php and is used to interact with the Pi_DB database.

HTML

Html is a standard markup language used to create web pages and its elements help form the building block of all websites. It has tremendously helped design the web interface built for Pi-Surveillance and interacts with other languages such as PHP, CSS and JavaScript to deliver the most effective service for users.

CSS

CSS is a simple styling language that can interact and structure HTML language when pointed to. It has been included in almost every page of the design for the web application developed and offers a more intuitive system for the user.

JavaScript

JavaScript (JS) is a lightweight, interpreted, programming language with first-class functions [9]. It is often refered to as the scripting language of the web. This can also be seen in most pages throughout the design of Pi-Surveillances web application.

MySQL

MySQL is a widely used open source database. It’s reliable, intuitive, and generally has no performance issues. It has been set up with Pi-surveillance web and android application so that users and admins can view, edit and add user settings along with camera and media addresses.

Android Studio (IDE)

After researching development software, and evaluation between android studio and eclipse, it was decided that android studio was most suited. From previous experience, eclipse is unreliable and experiences many functional issues. Android studio was designed specifically for android development and personally enjoyed the idea of working with a new IDE. Figure 13 below supports why android studio is the most feasible IDE to use.

Feature	Android Studio	ADT
Build Systems	Gradle	Ant
Maven-based build dependencies	Yes	No
Build varients and multiple-APK generation	Yes	No
Advanced Android code completion and refactoring	Yes	No
Graphical layout editor	Yes	Yes
APK signing and keystroke management	Yes	Yes
NDK support	Yes	Yes

Figure 13: Android Vs Eclipse

Bootstrap

Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first web sites.

Motion (Daemon)

Motion is an open source program developed in C which monitors the video signal from one or more cameras. It can detect if a significant part of a picture has changed (ie. motion). Motion uses a technique known as visual motion detection (VMD) to compare a series of sequential camera frames for differences at a pixel level. A change between a series of sequential frames is an indication of movement [10]. Motion is specifically designed for the Linux OS. It can also be used to feed events to a MySQL database. Motion also supports features such as zoom and media storage capabilities. Push notifications can also be configured when motion is detected.

Postfix

Postfix is used to route mail to other mail servers like Gmail (SMTP). This is used with a mail user agent called Heirloom mailx. It is also configured with motions configuration file by piping the mail client after an image is detected.

Heiloom Mailx

Using Heirloom mailx which is a mail user agent for UNIX systems. This will be the local mail client for my project that works with postfix to send mail to my Gmail account. A typical example of using Heirloom along with postfix might be: **echo ‘hi’ | mail –s Test <Email address>**

Putty

Putty is a free implementation of SSH and Telnet for Windows and UNIX platforms [11]. Used mainly to SSH into the server and develop the configuration and relevant files needed to complete the project.

ServoBlaster

This is software for the RaspberryPi, which provides an interface to drive multiple servos via the GPIO pins [12]. Using a breadboard, cables and battery pack, the servo blaster software can be used to test the pan tilt zoom bracket. Motion provides a tracking algorithm for the |servo motors and can be configured in the motion.conf file.

000webhost (Web + MySQL) & No-IP (Dynamic Domain Service)

Note: this technology was not implemented into the system but had initially intended to be of use.

Original Intended Purpose: a web application using HTML and PHP through the webhost service 000webhost. The web application will accept an IP and port number input by the user, and return a live stream of their IP cam/server. A MySQL database to store user credentials. It is also a free service and offers dynamic addressing to users.

Other Technologies

IPCamLive is a cloud based video broadcasting solution for IP cameras which embeds live video streaming to any webpage [13]. It’s an alternative for IP cameras which do not have a video player component. The video player provided by IPCamLive is based off of flash and HTML5 and can be used on multiple platforms which include PCs, Mobiles, tablets and MACs to name a few. It uses HTTP aligned with RTSP to stream the video which can display MJPEG, MPEG4 and H.264 streams.

Once a user registers and signs in, he/she can add a camera given the cameras URL as reference. Port forwarding also has to be set up in order to view video streams. IPCamLive provides html snippets of the video streaming which gives access to multiple users at any given time.

Once a user registers and signs in, he/she can add a camera given the cameras URL as reference. Port forwarding also has to be set up in order to view video streams. IPCamLive provides html snippets of the video streaming which gives access to multiple users at any given time.

The finished system consisted of similar features. Instead users can sign in given their IP and port number. This will give immediate access to the camera in which they wish to view.

System Requirements

System Requirements

Software Requirement	**Android Studio**
Software Requirement	**Motion**
Software Requirement	**Apache Web Server**
Software Requirement	**MySQL**
Hardware Requirement	**Raspberry Pi Model 2B**
Hardware Requirement	**Android Smart Phone**
Hardware Requirement	**PC**
Hardware Requirement	**Pan Tilt Zoom Bracket**
Hardware Requirement	**Servo motors 4.5v**
Hardware Requirement	**IP Camera**
Hardware Requirement	**External Drive**

Figure 14: System Requirements

Conclusion

Having not only evaluated similar systems in chapter 2 that relate to Pi-Surveillance but now also examined and evaluated the possible technologies to be used, the next chapter can discuss exactly what design methods are required.

The next chapter will provide a full overview on the system design and will consist of detailed diagrams outlining the important components used throughout the system.

Chapter 5: Design

Introduction

The design chapter gives a detailed look at the overall system architecture of Pi-Surveillance. The chapter includes a detailed diagram on the technical architecture used including a use case and entity relationship diagram of the database.

A section is provided on the Approach and methodology used throughout the system development process.

Finally the chapter finishes with an overview of the prototypes developed from the start of the project and are contrasted with the final system components.

Approach and Methodology

The incremental methodology approach has been considered in order to develop the apposed system. The reason for this choice is the ability to control and not over engineer or add flexibility unless needed. Similar to following the waterfall module, the incremental approach is divided into smaller, more easily managed modules. User requirements are suspended when an increment is under construction. This gives the opportunity for other increments to develop and provide a greater sense of flexibility for the changing requirements. Requirements that contain the highest of priorities are considered and developed first. This gives an advantage over alternative methodologies as it guarantees a quality over quantity effect.

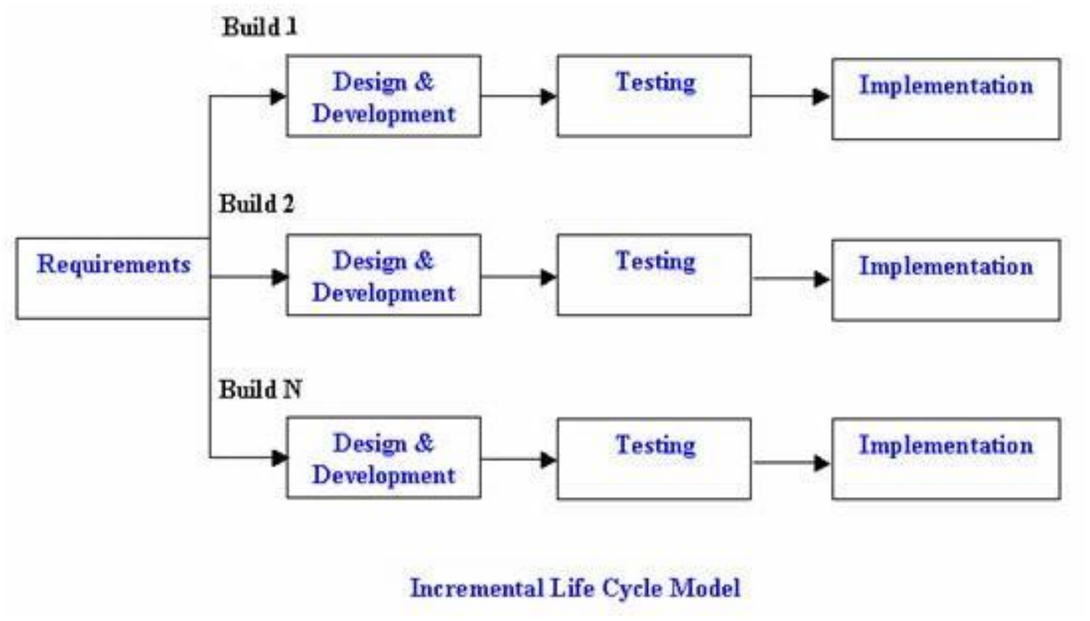


Figure 15: Incremental Approach

Architectural Design

The following diagram figure 16, is based on the main architecture of the system. Designed to give a more simple understanding of the system and display how it's put together. The presentation layer consists of both the android application and the web application. Each application is briefly examined on how it works and how it's developed. The application layer describes the setup of the server and how it works. Most of the configuration such as motion detection and sent notification is produced here. Finally the data layer briefly describes the database and what was used to develop it.

Presentation Layer	**Android:** ➢ Platform 6.0 – Target SDK 22 ➢ Connect to server over local + internet connection ➢ Displays both media and camera view **Web:** ➢ Developed using notepad ++ ➢ Displays both media and camera view ➢ Add, Update, view tables from MySQL ➢ Connects to server over local + public
Application Layer	**Server:** ➢ Built on Raspbian Jessie ➢ Connected to a router via Ethernet or wireless dongle ➢ Runs apache/MySQL ➢ Runs motion detection software ➢ Stores media files to external drive ➢ Pushes email notifications SMTP
Data Layer	**Database:** ➢ Created using MySQL service on Raspbian Jessie ➢ Handles INSERTS, SELECTS, UPDATES and DELETES. ➢ Consists of the users, camera, media and user_level table

Figure 16: Technical Archictecture

Other Design Documents

Use Case Diagram

The following use case diagram exhibits how users/admins interact with the system. Users and admins are first prompted with login access to both the android and web application. The user settings are first verified before continuing. Users can only view media, view camera and logout and login whereas admins have permissions to update users, update camera addresses and update media addresses.

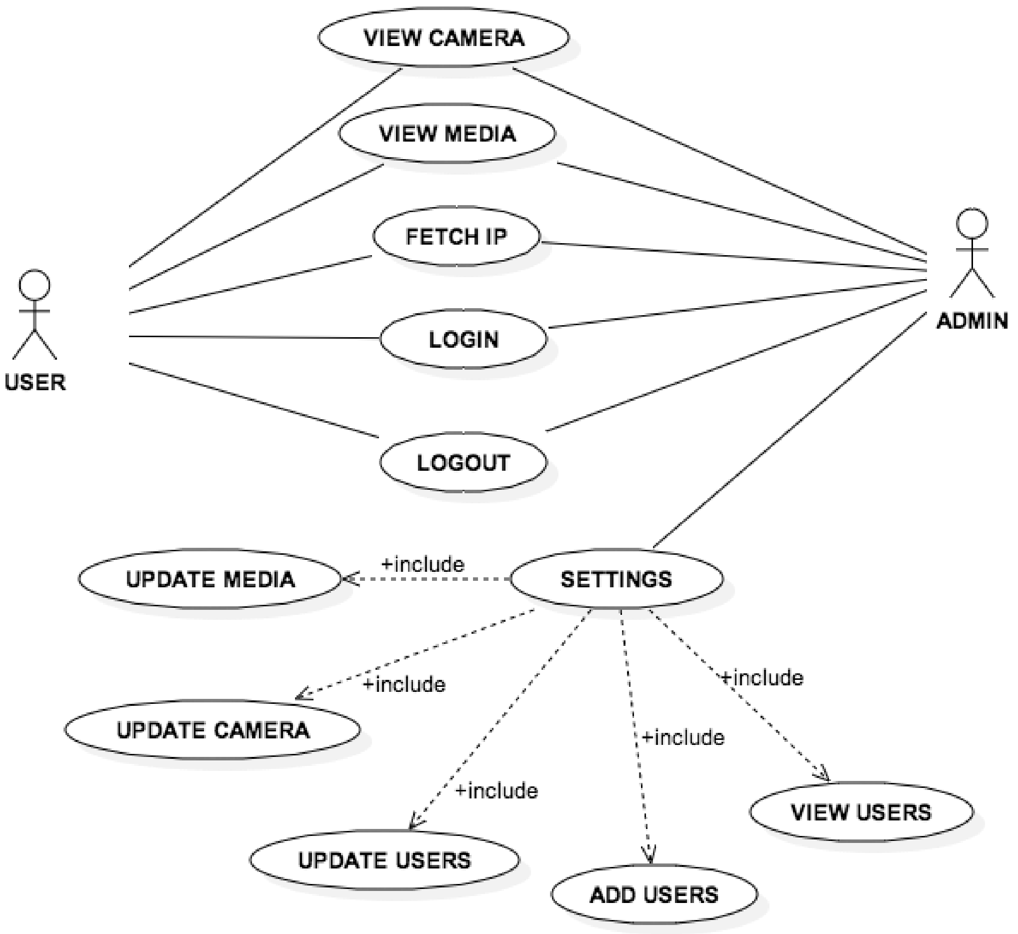


Figure 17: Use Case Diagram

Entity Relationship Diagram

The entity relationship represents the design used for the database. The users table consists of the users credentials. Users will be referenced by ID to assign a login system. The camera table consists of the current address of the camera. The media table consists of the current address of the media server. Finally, the user level table consists of the type of user that are saved to the database. It joins the users table by id. Level 1 grants user permissions whereas level 2 grants admin permissions.

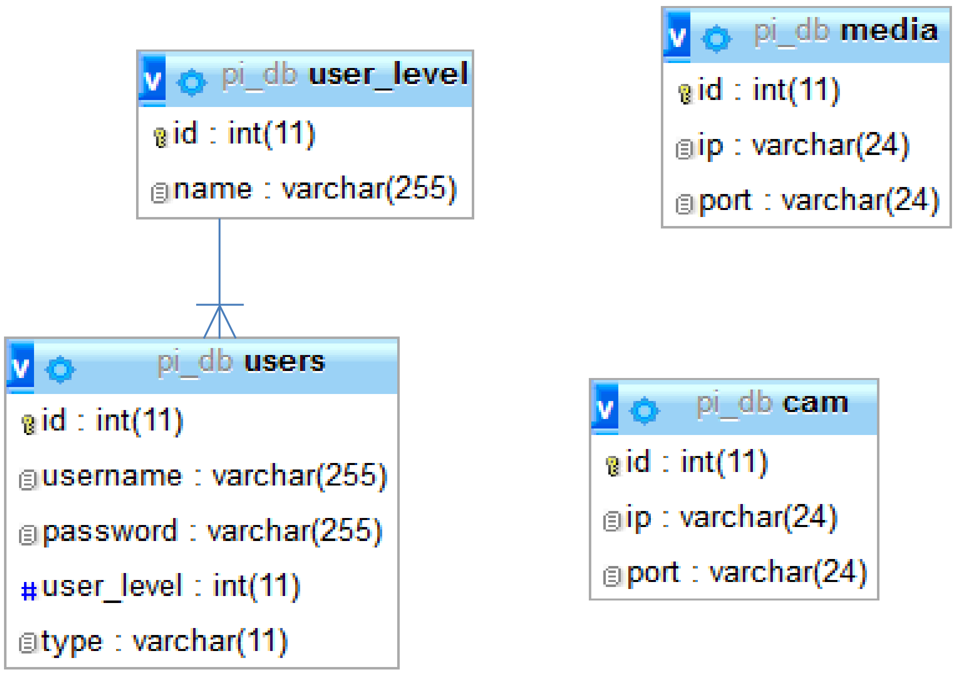
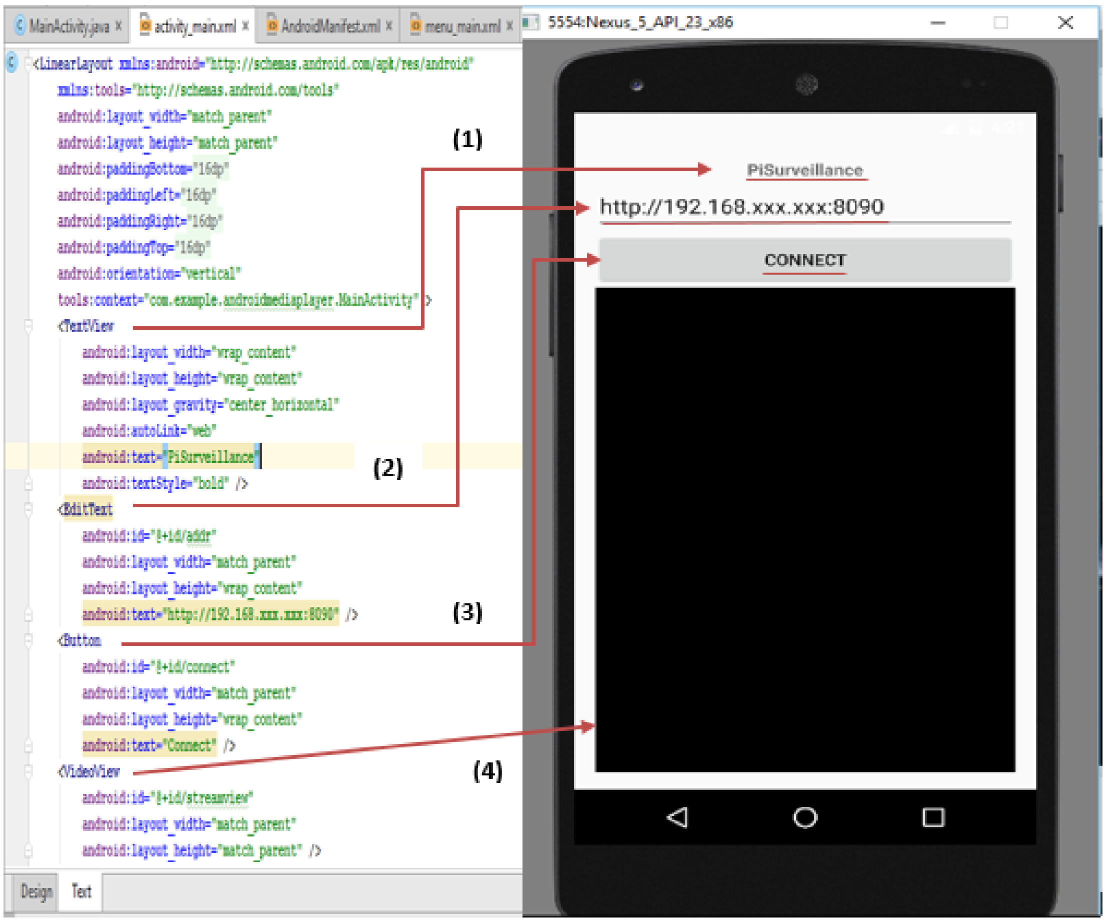


Figure 18: Entity Relationship Diagram

Prototyping and Development

The following section compares development prototypes achieved in semester 1 of college. The chapter proceeds to compare the prototypes against the final system components including the android application, web application and other ideas opposed. To begin the chapter will first describe the android application prototype. The final android application will then be addressed and evaluated.

The chapter will then look at the web application prototype, which will also be compared to the final application.



```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:orientation="vertical"
    tools:context="com.example.androidmediaplayer.MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:autoLink="web"
        android:text="PiSurveillance"
        android:textStyle="bold" />
    <EditText
        android:id="@+id/addr"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="http://192.168.xxx.xxx:8090" />
    <Button
        android:id="@+id/connect"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Connect" />
    <VideoView
        android:id="@+id/streamview"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

```
http://192.168.xxx.xxx:8090
```

Figure 19: Android Applicatio Prototype

The android application above was developed to target SDKs of versions 22. It did not consist of many activities and had only the main screen developed. It did however display the camera feed from server when a connect button is pressed. Users could enter there IP and port number as shown in figure 19. The android application uses HTTP to view image files that are refreshed on the server every 2 seconds. Users could view the stream locally or over public network.

When a user is happy with his/her IP address and port number, they could process the feed from the camera by hitting connect button.

As shown the xml file used has a linear layout. The video views height and width is set to match parent to guarantee a correct positon of display to the user. Using the import android.net.Uri, the connect button would then call the play stream function.

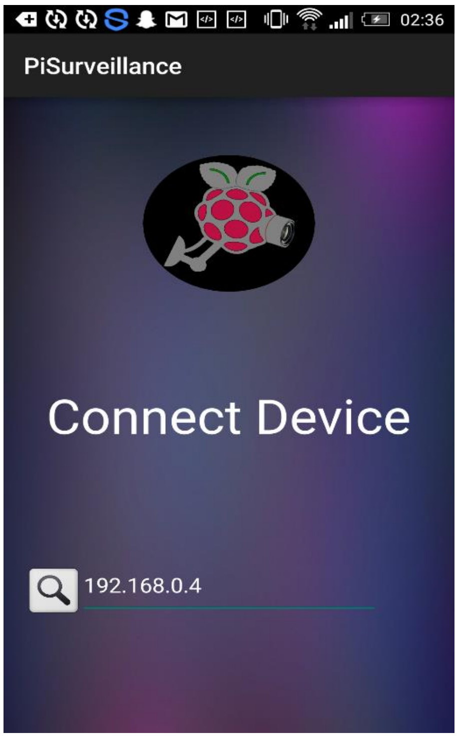


Figure 20: Fetch Ip

Fetch IP

The fetch IP activity is the first screen that the user is shown. The user is required to enter his/her target address. The address entered has an input type of ‘phone’ so only numbers are accepted. Once the address is entered and the search address button is pressed, the address is sent to a URL request activity. The address is then merged with a hardcoded string - :8090/Pi/index.php.

For example, 192.168.0.4/ then the URL :8090/pi/index.php. If the address is found, this returns a pass message and the database is available for user logins.

This then passes the user on to a login activity screen where they can attempt to login if their credentials are saved to the MySQL database.

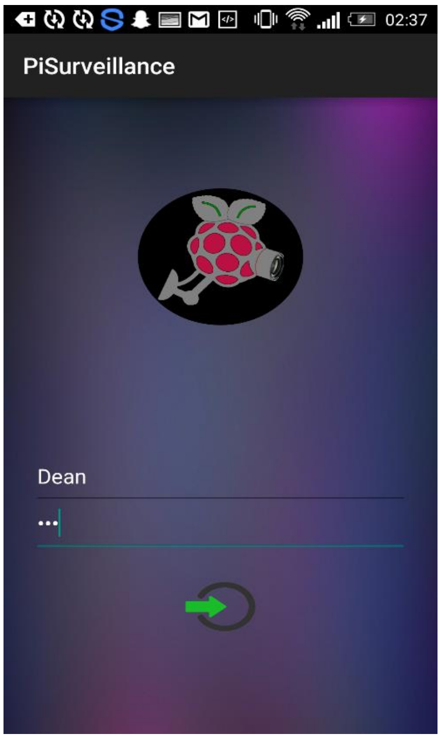


Figure 21: Login

User Login

The login activity prompts users and admins for their credentials saved to MySQL database. Once the user enters their credentials and clicks the process button, the username and password entered is posted to php script using JSON. JSON searches for a Boolean type called ‘successful which holds the SQL containing all user details within the users table. The statement is check for a username and password that both match a user.

If the credentials are invalid, a dialog box is shown that the credentials are invalid and the user may try again. However if the user details are valid, a login function is called and a user session is started.

The next activity displayed is camera view. Users can logout at any time.

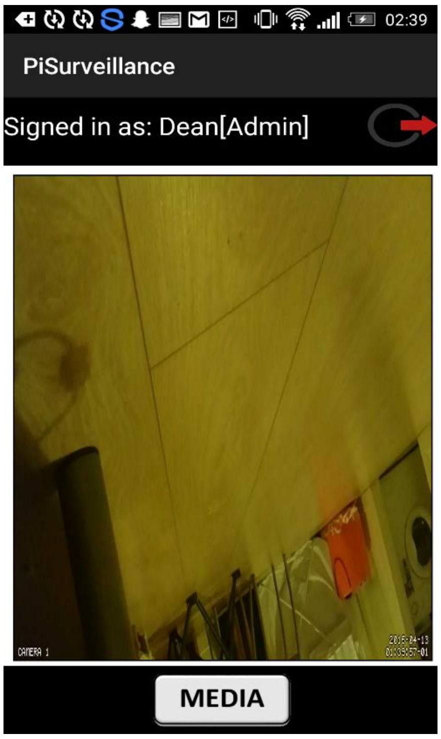


Figure 22: View Camera

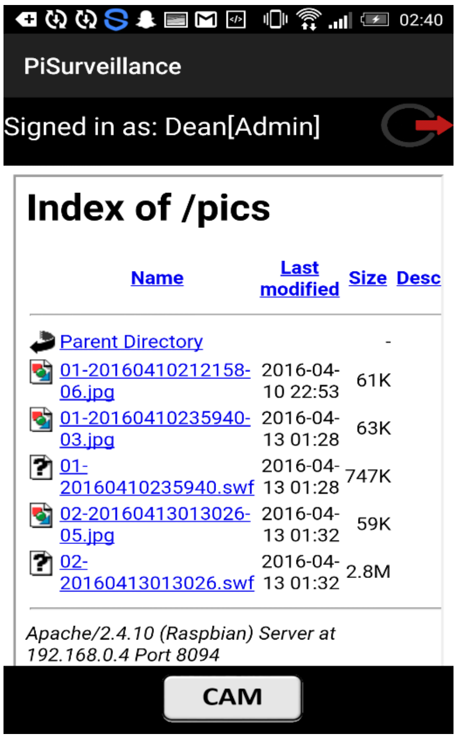
Camera View

The camera view activity contains a view of the live feed streaming from the server. This is achieved by including a html image within the activity.

The camera view xml layout consist of a header and footer. The header provides a logout button while displaying to the user or admin a welcome message with his/her name. This is achieved by using setters and getters provided that JSON has already returned the users details.

A media button is positioned in the footer which is linked to the media view activity. When users press the media button, the current activity is destroyed, carrying the user details to the next activity using the getters and setters.

The next activity is the media activity.



Index of /pics

Name	Last modified	Size	Desc
Parent Directory		-	
01-20160410212158-06.jpg	2016-04-10 22:53	61K	
01-20160410235940-03.jpg	2016-04-13 01:28	63K	
01-20160410235940.swf	2016-04-13 01:28	747K	
02-20160413013026-05.jpg	2016-04-13 01:32	59K	
02-20160413013026.swf	2016-04-13 01:32	2.8M	

Apache/2.4.10 (Raspbian) Server at 192.168.0.4 Port 8094

Figure 23: View Media

Media View

The media activity is quiet similar to the camera view activity. It consists of the same layout, which provide the same welcome message and logout button. However, the html found within the string is slightly changed. Because the information been provided is not an image, the information needs another way to be viewed. An html iframe is used to display a scrollable window to list the available media.

The footer remains the same except instead of providing the user with option to view media, they can view a new activity of the camera feed.

If the buttons is clicked, it destroys the current activity and passes on the users credentials using the getters and setters methods.

To conclude, Figures 20 – 23 above show the final design of the android application. Figure 20 is the fetch IP screen. Users can enter the IP of the address at which Pi-Surveillance is set up. Once connected, the login activity with load. The login screen prompts the user or admin for their details in order to access the camera and media from the server. If the user credentials are incorrect, a toast screen is displayed stating bad user credentials. Otherwise if the user credentials are valid, the user/admin is brought the camera stream activity figure 23. A message is displayed welcoming the user with his/her user level. Users can switch between the media and cam activity by selecting the buttons at the bottom of the screen. User can logout as they please on either screen.

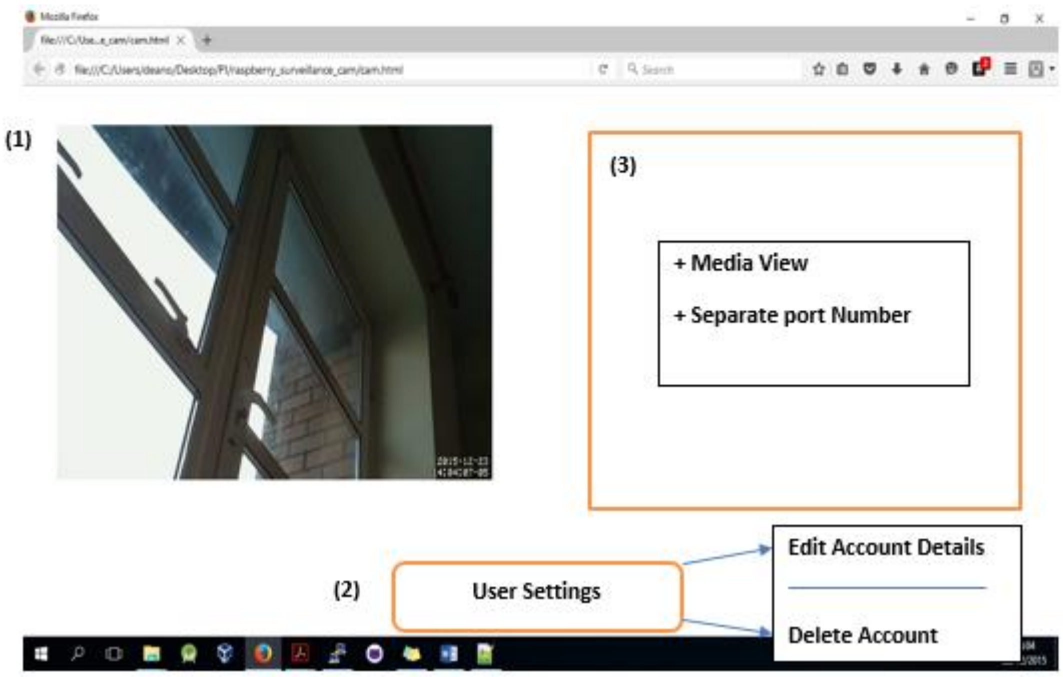


Figure 24: Web Application Prototype

Figure 24 above, shows the development of the web application at first attempt. There was not enough time to build a functional web application so a simple view was made including the camera stream on a clear html page. The image was edited so to initialise a clear outline on what was expected of the final web application.

(2) The user settings button intended to provide users the ability to edit credentials such as their name or port number of the stream etc. It was meant to offer users the option to delete their account if they no longer wish to use the applications. This implied that user permission will also be set up with the MySQL database.
(3) It was also intended that the media view will be provided over FTP using XBMC as the platform for the view.

Login

The login page below is the first page displayed as users attempt to access the camera feed. Similarly with the android application, a PHP file is created and contains a configuration set up calling the database. A separate PHP file is used to pass the users details to the database. This is achieved by using an html form, which passes the information to a post [isset] function created by PHP.

The page layout is made up of mostly CSS. This includes the header, footer, about button and the login form.

The username and password entered are checked using a select statement within the PHP file. If the username and password are not found in the returned values, a toast box is shown asking the user to 'retry'.

If the user is successful at logging in, a login session is started within another php file. They are then redirected to another php page where the camera view and media view is accessible.

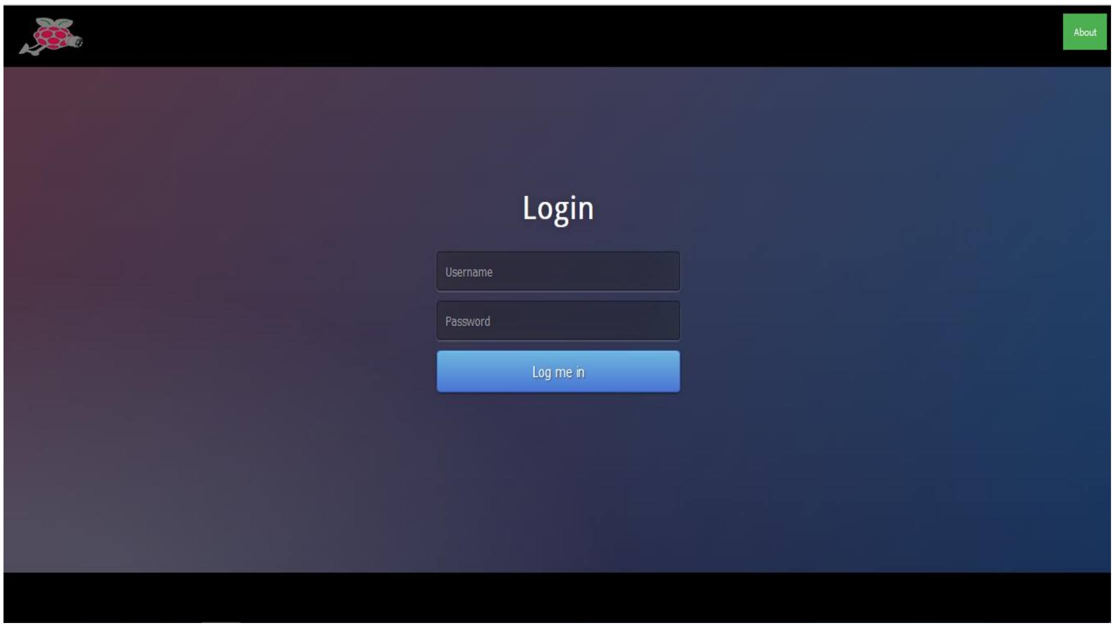


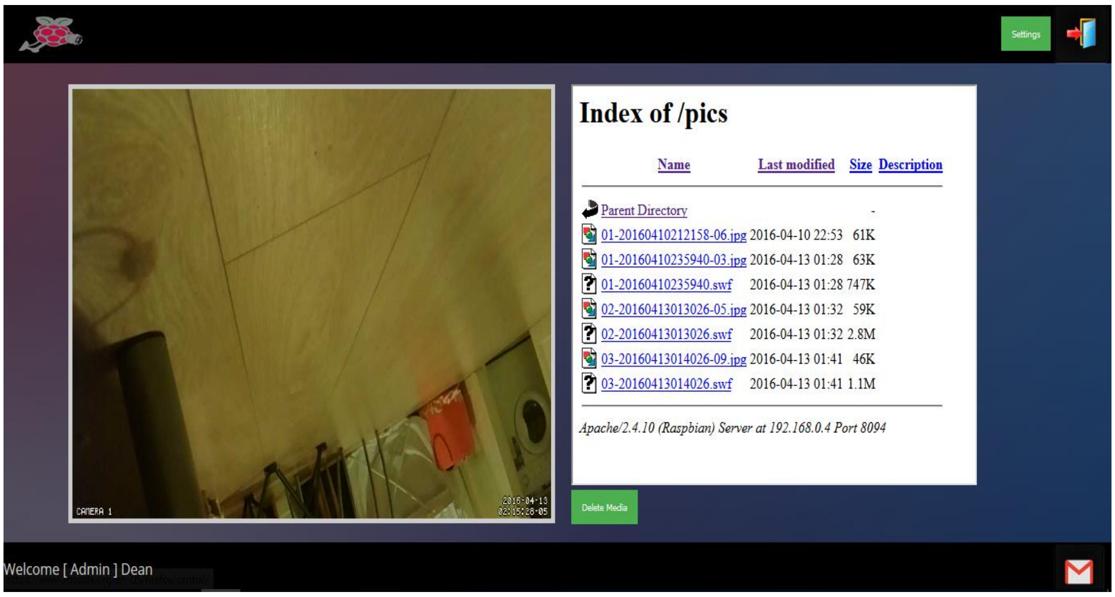
Figure 25: Web Login

Camera View

The camera view is the next page the user is directed to. The pages consists of the same CSS as the login screen. A change has however been made the footer which displays a welcome message.

The web application is set up so that there can be user permissions set between the admin and users. Figure 26 shows an example of an admin logged in. A settings button is displayed for admins that offers a dropdown menu. The menu lists a number of different options including edit users, add users, change media, and change camera. Note that these settings are not available to an ordinary user.

A button is also displayed for admins that allow them to quickly delete media if they feel there is too much media saved on the device. It should be mentioned that a script is also set up on the sever to delete all media every Sunday of each week. This means the user does not have to worry about checking how full the device is.



Index of /pics

Name	Last modified	Size	Description
Parent Directory		-	
01-20160410212158-06.jpg	2016-04-10 22:53	61K	
01-20160410235940-03.jpg	2016-04-13 01:28	63K	
01-20160410235940.swf	2016-04-13 01:28	747K	
02-20160413013026-05.jpg	2016-04-13 01:32	59K	
02-20160413013026.swf	2016-04-13 01:32	2.8M	
03-20160413014026-09.jpg	2016-04-13 01:41	46K	
03-20160413014026.swf	2016-04-13 01:41	1.1M	

Apache/2.4.10 (Raspbian) Server at 192.168.0.4 Port 8094

Figure 26: Caamera View

Add Users

The add users layout remains the same to the previous layouts. Admins can access the page using the drop down settings button.

Just like the login screen, a form is supplied using html. Note that an id for the user isn't provided because the user ids are set to auto increment. While admins can add users, the password of each user is encrypted using md5 check sum. Once the admin has filled the form and is pleased with the new user details, they can click the add user button which calls an insert statement within PHP.

The new user credentials are posted and passed through an if statement within PHP. If the user credentials do not match the input types relating to the database, the values are rejected and a toast message appears that they are bad values. If the values are true, an insert is passed and a new user is added.

Admins are redirected to the home screen where they can view the user table using the settings menu again.

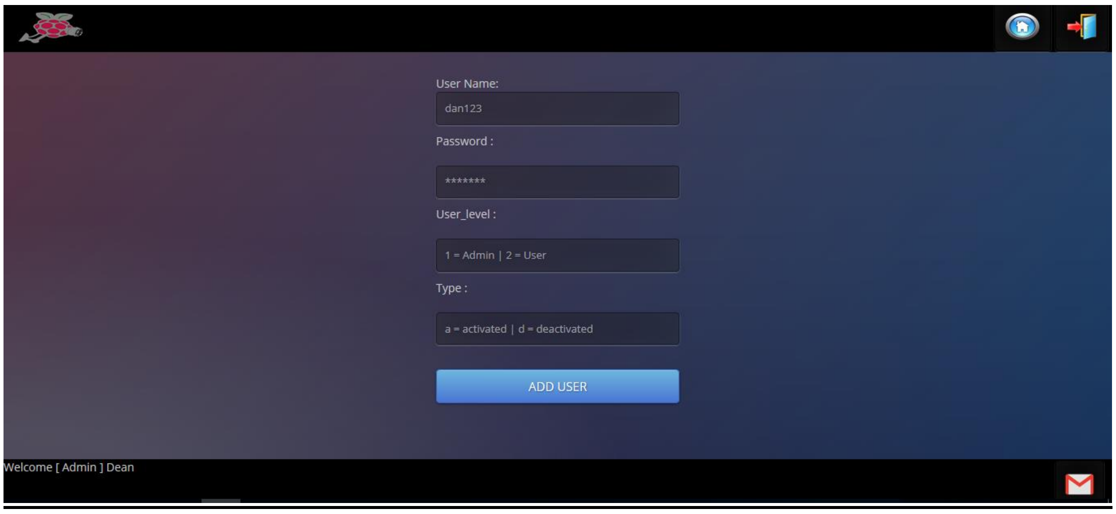


Figure 27: Add Users

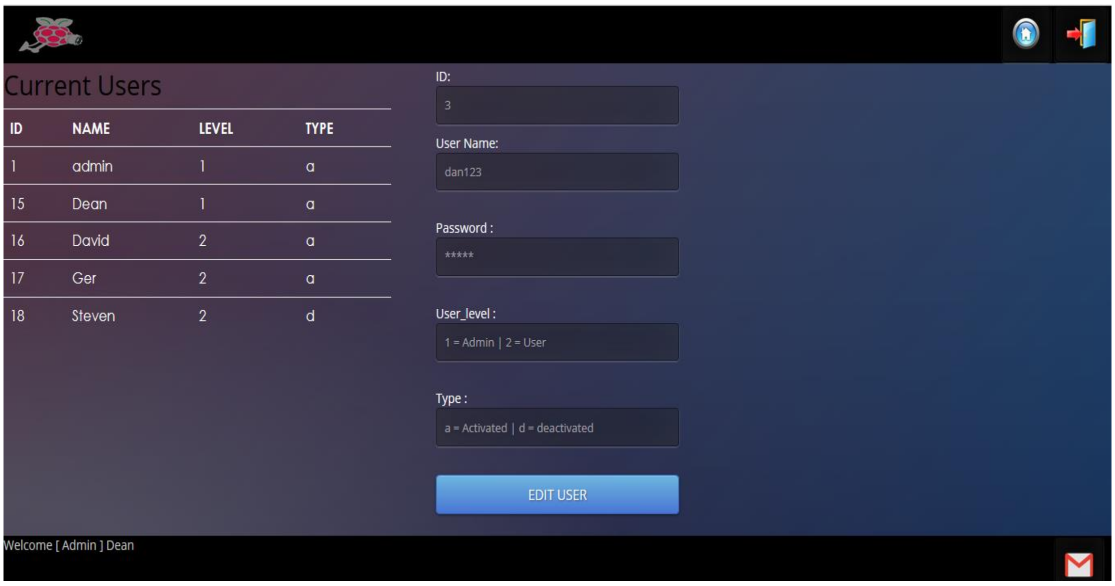
Edit Users

The edit user's page can be access via the drop down settings menu. It remains the same layout as the login and camera view page except a new home screen button is provided in case the user wishes to return to the camera and media view.

Admins can view what users are currently stored in the database. They may wish to update currents user credentials such as the username, password, user level and type of account the user has. This is achieved by selecting the user id and calling an UPDATE on the user id selected. The user level is based on what permissions the user has, whether he/she is an admin or user. This is achieved by inserting **1** for an admin or **2** for a user. The user type is then used to decide what type of account the admin wants to give the user. Admins can activate the account by inserting **a** or deactivate the account the account by inserting a **d**.

When the user selects the edit button, the information added from the admin is passed to a post method just like the login page. If a field remains empty, an error message is displayed using a toast screen and the admin is asked to retry. If all fields contain values, and each field suits the given type that relates to the database types, and update is successfully processed and the table updates after 2 seconds.

It should be mentioned that the table provided has been created using bootstrap. Bootstrap gives the table a more pleasant look.



Current Users

ID	NAME	LEVEL	TYPE
1	admin	1	a
15	Dean	1	a
16	David	2	a
17	Ger	2	a
18	Steven	2	d

ID: 3

User Name: dan123

Password : *****

User_level : 1 = Admin | 2 = User

Type : a = Activated | d = deactivated

EDIT USER

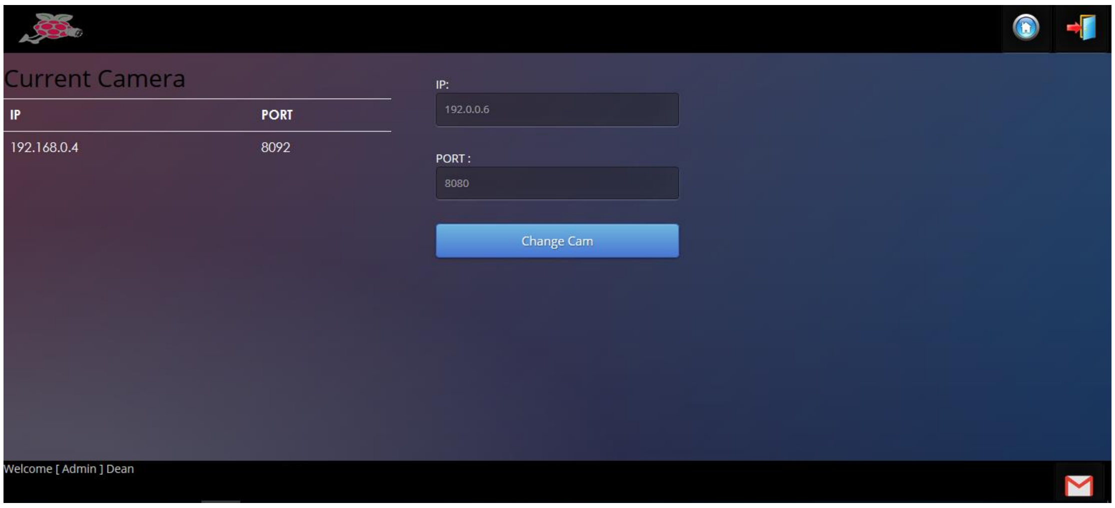
Welcome [Admin] Dean

Figure 28: Edit Users

Change Camera

The change camera page also remains the same layout as the other pages. Similarly to the edit users page, admins can view the current camera address of that the stream is leading from. Admins can update the camera if they want to add different addresses. The users of the system have no control of the camera settings and have to remain with the same camera view.

After the admin enters the new camera ip and port address, the page is refreshed and the new settings are put into place. The admin can return to the camera screen where the new settings are put into place and accessible. The same method applies to the change media page.



Current Camera

IP	PORT
192.168.0.4	8092

IP: 192.0.0.6

PORT : 8080

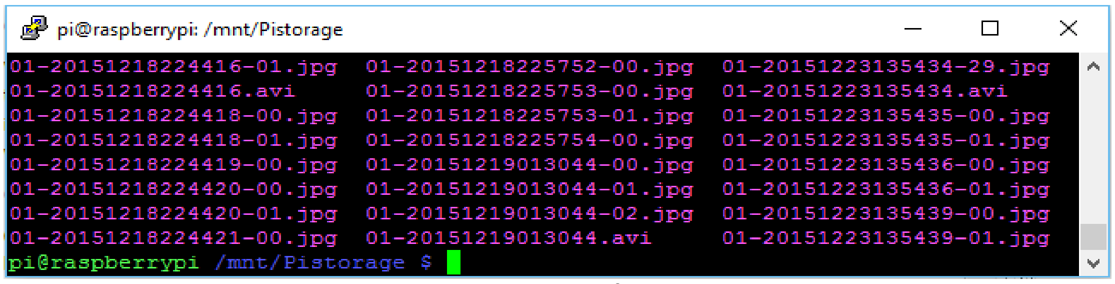
Change Cam

Welcome [Admin] Dean

Figure 29: Change Camera

To conclude, Figures 25 – 29 above show the final design of the web application. There is no need for a login screen as an IP address is already assigned to the web server. The first screen displayed is the login screen. The login screen prompts the user or admin for their details in order to access the camera and media from the server. If the user credentials are incorrect, a toast screen is displayed stating bad user credentials. Otherwise if the user credentials are valid, the user/admin is brought the camera stream page figure 25. For this example an admin was logged in. The figure provides the admin with a settings bar and delete media button that cannot be seen if an ordinary user is logged in. A message is displayed in the footer welcoming the user with his/her user level. Admins can navigate through the dropdown settings menu where a list of options are provided like add user, delete user, update etc. In figure 28 and 29 we see an example of the edit users and change camera page. Users can also logout at any time by hitting the logout button located at the top right corner of the screen.

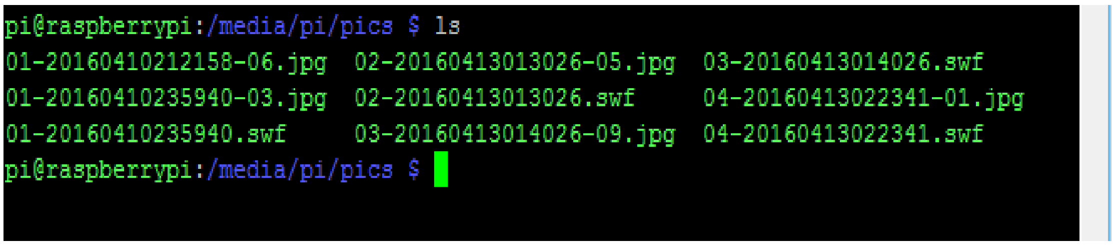
Another feature that had changed a huge part of the system was the ability to view and play media over http instead of downloading the files images. The original prototype sever had motion configured so that it would save media as jpg and .avi format. All images that picked up motion detection were saved to the external media device where they were then submerged into a .avi video file.



```
pi@raspberrypi: /mnt/Pistorage
01-20151218224416-01.jpg  01-20151218225752-00.jpg  01-20151223135434-29.jpg
01-20151218224416.avi     01-20151218225753-00.jpg  01-20151223135434.avi
01-20151218224418-00.jpg  01-20151218225753-01.jpg  01-20151223135435-00.jpg
01-20151218224418-01.jpg  01-20151218225754-00.jpg  01-20151223135435-01.jpg
01-20151218224419-00.jpg  01-20151219013044-00.jpg  01-20151223135436-00.jpg
01-20151218224420-00.jpg  01-20151219013044-01.jpg  01-20151223135436-01.jpg
01-20151218224420-01.jpg  01-20151219013044-02.jpg  01-20151223135439-00.jpg
01-20151218224421-00.jpg  01-20151219013044.avi     01-20151223135439-01.jpg
pi@raspberrypi /mnt/Pistorage $
```

Figure 30: Saved motion files Prototype

After configuring motion to save .swf files instead, this meant that users could view the videos over the browser instead of having to download the files. Also, instead of capturing every image with motion, a configuration was set so that only the first image would be saved along with the video following it as shown in figure 31 below.



```
pi@raspberrypi:/media/pi/pics $ ls
01-20160410212158-06.jpg  02-20160413013026-05.jpg  03-20160413014026.swf
01-20160410235940-03.jpg  02-20160413013026.swf     04-20160413022341-01.jpg
01-20160410235940.swf     03-20160413014026-09.jpg  04-20160413022341.swf
pi@raspberrypi:/media/pi/pics $
```

Figure 31: Motion Saved Media Final

Chapter 6: Implementation

Introduction

This chapter focuses on techniques used to join the main components of the system together. It discusses the main areas of functionality throughout the system

Presentation Layer

The android application has been developed using android studio. The application was built on the android platform 6.0 Marshmallow targeted at devices with a minimum of SDK 8 and maximum 22. A HTC device was used for testing the application rather than the emulator which can be very slow.

The application will consist of 4 main pages: fetch IP, login, camera view and media view. It was originally planned for the system to be able to conduct a network scan if users don't know the IP or port but unfortunately time management became an issue and the idea was dropped.

The fetch IP was built so that if the user was home or away, they were able to locate the server by fetching the local/public IP first.
The login page will first prompt the user to login with details that have already been set by the admin such as a saved username and password.
Once the user has logged in, they will be redirected to the camera view page. This will provide users with the live video stream from their camera. The camera will be addressed by the IP and port number that is saved to the MySQL database on the server. The activity also consists of buttons such as media and logout. The user name of the user is displayed if the user I successful upon login. The media page is quiet similar except instead of a view of the camera, it displays the saved media on the device.

Similarly like the android application, the web application provides users with a login screen. Note that when the system is purchased, a generic admin username and password is supplied so that the user can add a new admin if he or she pleases.
After the user has logged in, they can view the camera live feed as well as the media view. There is also a logout button to provide a user session destroy. Similarly to the android app, the web application provides a username of the user if they are successful at logging in.

If an admin is logged in, they are provided with an extra settings menu and delete media button. The settings menu leads to different pages like edit user details, change camera address, change media address and add user.

Application Layer

The development of the Pi server consists of bash scripts and configuration files. It runs various software technologies including apache2, motion and MySQL. The server will always require a network connection in order to deliver its purpose. A Pi camera is connected to the Pi server and provides video stream for the android and web components. An external device is also be connected to the server to provide media storage for recorded videos/pictures.

The server functions off the operating system Raspbian Jessie. While the router runs off IPv4 settings, Port forwarding is enabled so the android device and web application can connect to the server through internet communication from any location. The server can also be given a dynamic domain address provided by No-Ip. This allows it to access the camera from anywhere even if the routers address settings changes. Upon boot of the server, a running daemon called motion is enabled. This provides a number of different features which are used with the Pi camera (ie. motion detection). Adding the external hard drives UUID (/etc/fstab folder) will automatically mount the device upon boot which gives users the privilege to save media to the device. Notifications are configured with motion using postfix as the SMTP and Heirloom mailx as the client email for the server. When motion is detected, an email is sent to the users email address notifying him/her. While the email is being sent, a video is also recorded based on the time the motion senses change in the images. Videos are pre captured by the 2 seconds and are saved directly to the external device. They can be viewed through the web and mobile application. A simple script is set up to delete video/images from the external every Sunday as shown in figure 34 below. Finally, the server operates two servo motors attached to a pan tilt zoom bracket which can track the motion detected within the image. An example of the setup is shown below in figures 32 and 33.

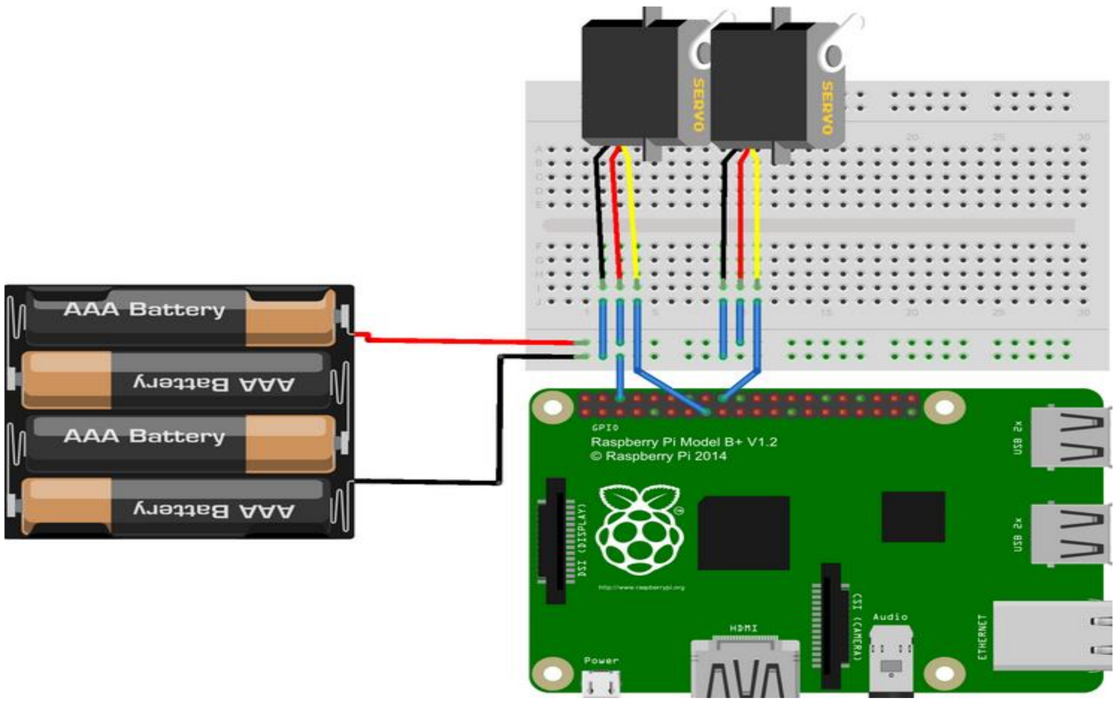


Figure 32: Fritzing

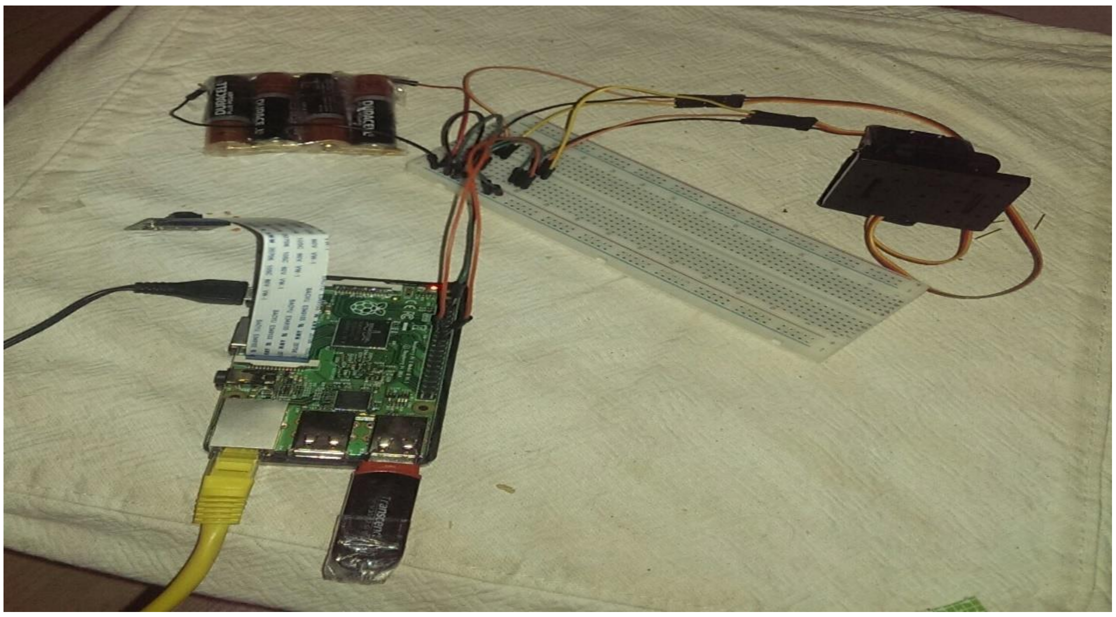
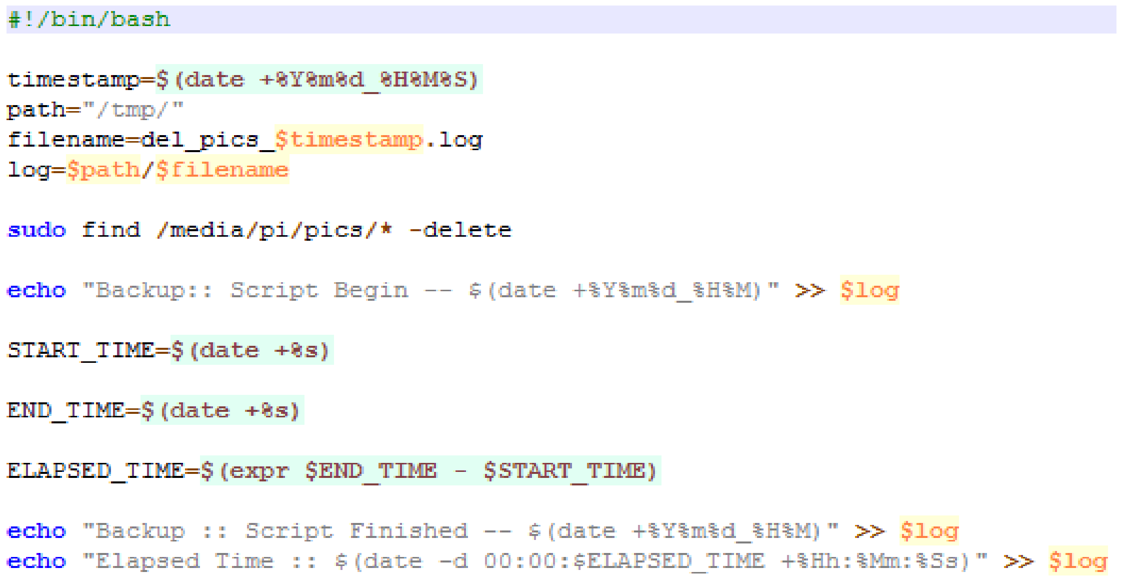


Figure 33: PTZ Final



```
#!/bin/bash

timestamp=$(date +%Y%m%d_%H%M%S)
path="/tmp/"
filename=del_pics_$timestamp.log
log=$path/$filename

sudo find /media/pi/pics/* -delete

echo "Backup:: Script Begin -- $(date +%Y%m%d_%H%M)" >> $log

START_TIME=$(date +%s)

END_TIME=$(date +%s)

ELAPSED_TIME=$(expr $END_TIME - $START_TIME)

echo "Backup :: Script Finished -- $(date +%Y%m%d_%H%M)" >> $log
echo "Elapsed Time :: $(date -d 00:00:$ELAPSED_TIME +%Hh:%Mm:%Ss)" >> $log
```

Figure 34: Delete Media Script

Data Layer

Lastly, the data layer is made up of a MySQL database stored on the raspberry pi. The database consists of user credentials, camera address settings, media address settings and user level settings.

The user level table relates to the user table and allows admins to add regular users or admins. Below are detailed diagram representations of the database and its components.

Chapter 7: Testing

Introduction

The following section describes the approach taken to test the functionality of the 3 system components: Android Application, Web Application and Pi Server.
Below are a number of test cases used to describe preconditions, expected results and post conditions that the system should produce.

ID: 1 Android	ID: 2 Android
Name: Logout	**Name:** Login
Purpose: Logout of the system when satisfied with session.	**Purpose:** Login with user credentials that have already been registered to the systems database
Procedure steps: ➢ Open android application ➢ Fetch IP address ➢ Login ➢ Logout	**Procedure steps:** ➢ Open android application ➢ Hit login button (Username + Password)
Expected Results: The user or admin should be able to logout after they are finishing viewing the camera and media windows. This protects a user's account and allows other users to login on the same device if needed.	**Expected Results:** After logging in with the users saved credentials, he/she should be able to view the live camera feed from his/her camera

ID: 3 Android	ID: 4 Android
Name: Camera View	**Name:** Media View
Purpose: Users can view live camera feed from the server on their device	**Purpose:** Users can view media that has been detected and saved to the external device
Procedure steps: ➢ Open android application ➢ Fetch IP ➢ Hit login ➢ View camera feed	**Procedure steps:** ➢ Open Android application ➢ Fetch IP ➢ Login ➢ View media
Expected Results: Users should be able to view camera feed after logging in	**Expected Results:** Users should be able to view media files after logging in

ID: 15 Android	ID: 15 Web
Name: Fetch IP	**Name:** Logout
Purpose: Users should be able to search for an address that the camera is located at	**Purpose:** Users should be able to logout and end a user session
Procedure steps: ➢ Open Android Application ➢ Search for IP address	**Procedure steps:** ➢ Open web application ➢ Login ➢ Logout
Expected Results: Users should be able to login provided they supply the correct address	**Expected Results:** Users should be able to logout and end a user session provided they have logged in with the correct user credentials

ID: 5 Web	ID: 6 Web
Name: Edit Users	**Name:** Add
Purpose: Admins should be able to update users	**Purpose:** admins should be able to add users
Procedure steps: ➢ Login ➢ Settings menu ➢ Edit users page ➢ Submit Form	**Procedure steps:** ➢ Login ➢ Settings menu ➢ Add users page ➢ Submit Form
Expected Results: Once a form has been submitted by the admin, he/she can view new update user	**Expected Results:** Once a form has been submitted by the admin, he/she can view new user

ID: 7 Web	ID: 8 Web
Name: Edit Cam	**Name:** Login
Purpose: Admins should be able to change camera addresses	**Purpose:** Login with user credentials that have already been added to the system
Procedure steps: ➢ Login ➢ Got to settings ➢ Update camera page	**Procedure steps:** ➢ Login
Expected Results: After admins log in to their accounts. They should be able to edit camera details located in the settings menu. This allows users to add other cameras to their system.	**Expected Results:** If the user has been already added to the user database, they should be able to login to view the camera and media files

ID: 9 Web	ID: 10 Web
Name: Edit Media	**Name:** Settings
Purpose: Users can change the address location of the viewed media	**Purpose:** The settings feature allows users to edit tables in MySQL such as add users, edit users, update camera and update media
Procedure steps: ➢ Login ➢ Locate settings menu ➢ Update the media address	**Procedure steps:** ➢ Login as Admin ➢ View settings bar
Expected Results: Users should be able to access a new media saved location provided they give the right address	**Expected Results:** The settings button opens a view which contains pages with submit forms like add user, edit user, update camera, and update media

ID: 13 Web	ID: 14 Server
Name: Media playback	**Name:** Motion
Purpose: Users can view media that they have previously recorded	**Purpose:** When the camera senses a change in an image, it begins to record a video until the change has normalized. (ie. motion)
Procedure steps: ➢ Navigate to web application ➢ Login ➢ Interact with media view (scroll)	**Procedure steps:** ➢ Power on server ➢ Run motion (daemon) ➢ Activate Pi camera ➢ Refresh stream frame
Expected Results: Once logged in, if users have previous saved videos, they can view it using the media view on the main page.	**Expected Results:** Videos should save to the specified file path given in the motion configuration file. This provides users with media playback.

ID: 15 Server
Name: Notifications
Purpose: Provide notifications to the users email to notify him/her when motion is detected
Procedure steps: ➢ Power on server ➢ Run motion (daemon) ➢ Activate Pi camera ➢ If motion detected: call function to send email using postfix daemon
Expected Results: Users should receive email notifying him/her about motion detected

ID: 15 Android
Name: Fetch IP
Purpose: Users should be able to search for an address that the camera is located at
Procedure steps: ➢ Open Android Application ➢ Search for IP address
Expected Results: Users should be able to login provided they supply the correct address

Chapter 8: Results and Evaluation

Introduction

This chapter takes a look at overall implementation of the final system. It discusses what has been achieved and what has yet to be completed. It examines the technologies that have been dropped since starting the development phase of the system and explores technologies that may be of use in the coming future.

If we look at the overall evaluation of the system. There has been a substantial change in the technologies used. There has been a swift change in the overall layout of the system and some features have and haven't be implemented as originally intended.

Omitted Requirements

As previously discussed, some technologies were either very time consuming, replaced or simply no longer needed for Pi-Surveillance.
For example the below test cases had to be removed from the system due to time consumption on the system.

ID: 11 Web	ID: 12 Web
Name: Edit	**Name:** Delete
Purpose: The edit button displayed on the settings menu gives user access to edit user credentials	**Purpose:** The delete button displayed on the settings menu gives users the ability to delete their accounts
Procedure steps: ➢ Navigate to web application ➢ Login ➢ Open settings ➢ Hit edit	**Procedure steps:** ➢ Navigate to web application ➢ Login ➢ Open settings ➢ Hit delete
Expected Results: Once users configure their new credentials, they are redirected to the login page so that the new credentials can be refreshed	**Expected Results:** Once users hit delete, they are prompted with a confirmation message. If 'yes' is chosen, the users account is deleted and they are redirected to the login page.

Another section that had to be adjusted since the development of the system was the technologies used. Below are two technologies that were included in the interim report submitted in December. They have since then been removed because of new replacement technologies.

<u>Proposed idea</u>
XBMC (Kodi) is a free and open source media player developed by the XBMC foundation [10]. Available on multiple operating systems, XBMC provides users with

a play and view feature of media saved locally. Media can also be viewed over the internet.

Replacement

This has been replaced with the simple mail transfer client. SMTP is easy to use and can be easily configured into motions configuration file. It has proven a lot less hassle for the system.

Proposed Idea

In order to stream and provide media playback, a plan was created to develop a web application using HTML and PHP through the webhost service 000webhost. The idea was that the web application will accept an IP and port number input by the user, and return a live stream of their IP cam/server. It was intended that the website would also be used to create a MySQL database to store user credentials. Webhost seemed favourable option at the time due to previous experience using the HTML, PHP and SQL languages.

Replacement

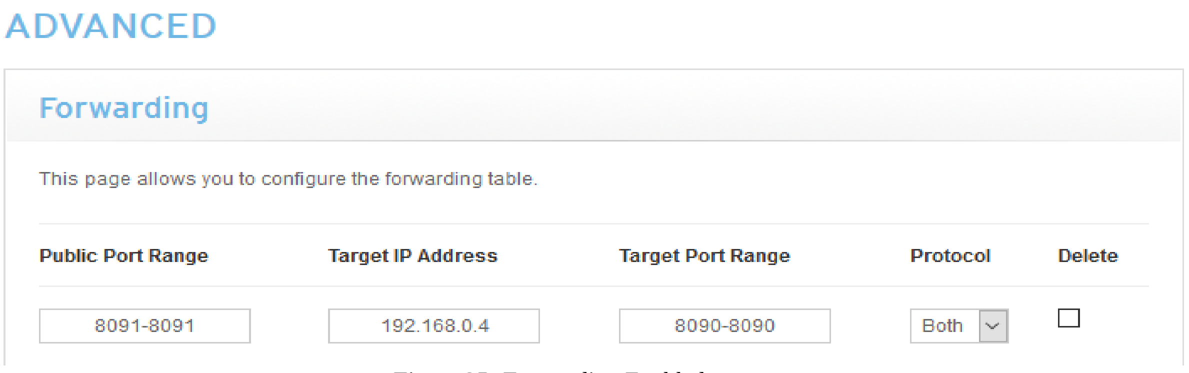
Instead of the above idea, a much simpler solution was available. The system had replaced this idea with a LAMP server. Creating a lamp server was not only more secure because of local storage but it also required a simple set up. A lamp server consists of an apache webserver, a MySQL database and the PHP scripting language.

Issues and Risks Faced

There were a number of different incidents and error reports throughout the development of the system.

IPv6 to IPv4

Due to the strict permissions of IPv6, in order to access the server and stream the media, the routers provider has to allow a switch a switch back to IPv4 settings. This granted admin access to the routers settings which meant one could enable port forwarding on the servers IP.



ADVANCED

Forwarding

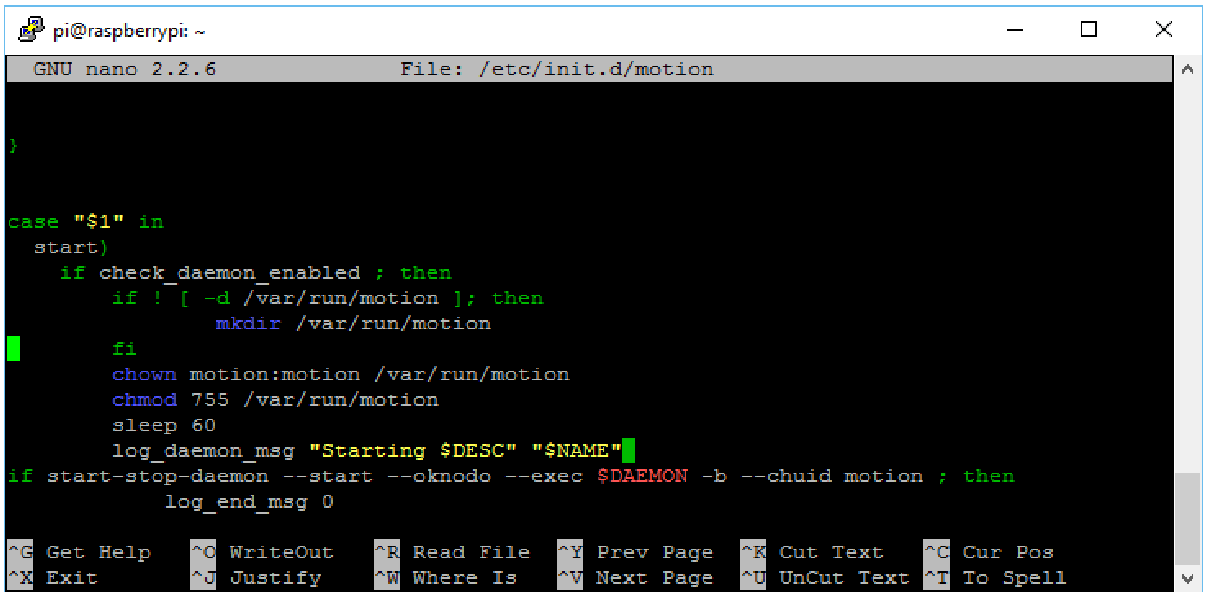
This page allows you to configure the forwarding table.

Public Port Range	Target IP Address	Target Port Range	Protocol	Delete
8091-8091	192.168.0.4	8090-8090	Both	☐

Figure 35: Forwarding Enabled

External Device

The external device that was connected to the server had had to be automatically mounted upon the system boot. Unfortunately problems were faced trying to configure motion to work with the device. Motion wouldn't start because it could not recognise that the external device was mounted upon the system boot. Thankfully, a solution found online suggested to add a sleep function to the daemon file /etc/init.d/motion. This meant that the raspberry pi could start and mount the external device in time for motion to begin and recognise the mounted device.



```
pi@raspberrypi: ~
  GNU nano 2.2.6              File: /etc/init.d/motion


}


case "$1" in
  start)
    if check_daemon_enabled ; then
        if ! [ -d /var/run/motion ]; then
                mkdir /var/run/motion
        fi
        chown motion:motion /var/run/motion
        chmod 755 /var/run/motion
        sleep 60
        log_daemon_msg "Starting $DESC" "$NAME"
if start-stop-daemon --start --oknodo --exec $DAEMON -b --chuid motion ; then
            log_end_msg 0

^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Figure 36: Error External Drive Directory

Dynamic Domain Address No-IP

Upon developing the web and android application, issues were experienced with connecting to the server outside the network. A solution suggested that if one had set up a dynamic domain address, it would enable me to access the server from anywhere. Enabling the address meant that the IP set to the server would remain the same and would not need changing.

Future Work

Wifi Enabled

A simple yet effective solution that should be incorporate into the system design is the ability to place the camera anywhere around the house provided it can reach the routers connection. WIFI dongles can be bought from sites like amazon and eBay and are not expensive.

Image Processing

A final feature which would also greatly benefit the security purposes of the system is face recognition. Open source libraries like OpenCV can be implemented into python or C scripts to help achieve this feature.

Chapter 9: Conclusion

The research undertaken throughout the report suggests great potential for the opposed system provided that the requirements displayed in chapter 3 are met. The main issues at hand is time management and needs to be dealt with accordingly. There is still a vast amount of research needed in order to complete the project in full which includes image processing as the main feature. Note that there is still also pre-existing areas of the system the need attending such as the development of the media server and its view, along with the possibility of the user interacting with the server. I have full intentions to include as many features into the systems as possibly. My main goal is to acquire enough information needed to apply face recognition. To conclude, having seen how much I have progressed and completed with regards to the beginning of the project and the features I have currently functioning, I am confident that I am capable of delivering the extra features that are needed to complete the system.

References

[1] The Best Security Inventions In History. 2016. *The Best Security Inventions In History*. [ONLINE] Available at: http://www.slideshare.net/ajsteeldoors/best-security-inventionshistory.
[Accessed 13 April 2016].

[2] *The history of CCTV - from 1942 to present | The latest news from the Computer and IT Industry | PC Retail*. [ONLINE] Available at: http://www.pcr-online.biz/news/read/the-history-of-cctv-from-1942-to-present/034658.
[Accessed 13 April 2016].

[3] 10 Heuristics for User Interface Design: Article by Jakob Nielsen. 2016. *10 Heuristics for User Interface Design: Article by Jakob Nielsen*. [ONLINE] Available at: http://www.nngroup.com/articles/ten-usability-heuristics/.
[Accessed 13 April 2016].

[4] 10 Heuristic Principles – Jakob Nielsen's ~ UXness . 2016. *10 Heuristic Principles – Jakob Nielsen's ~ UXness* . [ONLINE] Available at: http://www.uxness.in/2015/02/10-heuristic-principles-jakob-nielsens.html.
[Accessed 13 April 2016].

[5] *Android could represent over 50% of worldwide devices by 2016 - Market Realist*. [ONLINE] Available at: http://marketrealist.com/2014/04/android-represent-50-worldwide-devices-2015/.
[Accessed 13 April 2016].

[6] Extensible Markup Language (XML). 2016. *Extensible Markup Language (XML)*. [ONLINE] Available at: https://www.w3.org/XML/.
[Accessed 13 April 2016].

[7] PHP: What is PHP? - Manual . 2016. *PHP: What is PHP? - Manual* . [ONLINE] Available at: http://php.net/manual/en/intro-whatis.php.
[Accessed 13 April 2016].

[8] JavaScript Tutorial. 2016. *JavaScript Tutorial*. [ONLINE] Available at: http://www.w3schools.com/js/.
[Accessed 13 April 2016].

[9] *Remote Motion-Activated Web-Based Surveillance with Raspberry Pi | Programmatic Ponderings*. [ONLINE] Available at: https://programmaticponderings.wordpress.com/2013/01/01/remote-motion-activated-web-based-video-surveillance-with-raspberry-pi/.
[Accessed 13 April 2016].

[10]PuTTY: a free SSH and Telnet client. 2015. *PuTTY: a free SSH and Telnet client*. [ONLINE] Available at: http://www.chiark.greenend.org.uk/~sgtatham/putty/.#
[Accessed 23 December 2015].

[11]*PiBits/ServoBlaster at master · richardghirst/PiBits · GitHub*. [ONLINE] Available at: https://github.com/richardghirst/PiBits/tree/master/ServoBlaster.
[Accessed 13 April 2016].

[12]*Embedding IP camera live video stream in web page* (no date) Available at: https://ipcamlive.com/
[Accessed: 13 April 2016].