TECHNOLOGICAL UNIVERSITY OF DUBLIN

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

Bachelor of Science (Hons)

Networking Applications and Services

DT080B/TU819

Module: Project (PROJ4002)

Year: 2020-2021

TITLE: Odin's Eye

Supervisor: Kevin Tiernan

Student: Andrew Heaney

Date Submitted: 13 May 2021

Table of Contents

Declaration

I, the undersigned, declare that this report is entirely my own written work, except where otherwise accredited, and that it has not been submitted for a degree or other award to any other university or institution.

Signed: Andrew Heaney

Date: 13/05/2021

Abstract

The aim of this project is developing a camera system which provides the user with a subscription-less 24/7   surveillance system. Another aim of this project is to develop a system which can allow the video produced to be stored on the device. This will be one of the bigger challenges of the project as videos are known to take up a lot of space, especially when the video recorded is produced by a 24-hour camera service.

In this project, a camera system is developed. The project will be developed on a Raspberry Pi 4B Board, the camera is a Raspberry Pi High Quality camera. The language used for recording the video will be python, and an open-source module call “OpenCV” will also be used to aid this process.

The Raspberry Pi will also function as a webserver for the user to access the videos recorded. Apache2 will be the webserver software that will be used for the Website, and the primary languages used for the Website is HTML, CSS, and PHP.

Chapter 1: Introduction

In today’s world, security is a continuing issue. Big tech companies provide consumers with products and promise of 24-hour security camera video. These services operate often with a paid subscription. While they do offer a free service, the service is very minimal and don’t offer many features. The Aim of this project is to provide a subscription-less service, that provides an alternative to these products provided by big tech companies like Google or Amazon.

Another aim of this project is to provide consumers with the ability to stay in control of their own data. In today’s world, a person’s data is not as easy to maintain. This is because companies use that information and, in some situations, companies leak or sell users data.

This product will use Raspberry Pi as the main system for recording video and displaying said video on a website using apache2 as a webserver. The website uses forms in order to find which video should be viewed. This is done in order to organise the webpage and make sure it isn’t overwhelming to the user.

The website incorporates MariaDB as a way of storing users, this is done as a security precaution, so when a user signs in, they can’t just inspect the page to find the username and password. In order to utilise the Database, PHP is used to make queries. To ensure that injection attacks aren’t an issue, A filter is us

1.1 Milestones

- Develop System capable of recording suitable videos for a html Website, using Python, OpenCV and a suitable codec.
- Design a site suitable for users to interact with that can play recorded video, using HTML and CSS
- Implement some security features, using MariaDB and filters for login system.
- Implement Storage management features which is capable of keeping the memory usage under control, using Crontab, and OpenCV in order to implement motion control.

Chapter 2: Hardware Used

2.1 Raspberry Pi

For this project, the Raspberry Pi4B model was used. This model has 8Gb of ram, with a Quad Core arm processor at 1.5GHz. The Raspberry Pi 4B comes with a Gigabit ethernet port, which is what we will be using. It should also be noted that instead of using a DHCP connection the Connection will be static, so that the address doesn't change.

There is also a 2-lane MIPI CSI Camera Port which allows for the Camera to be connected directly to the GPU rather than taking up valuable processing power for the CPU.[12]

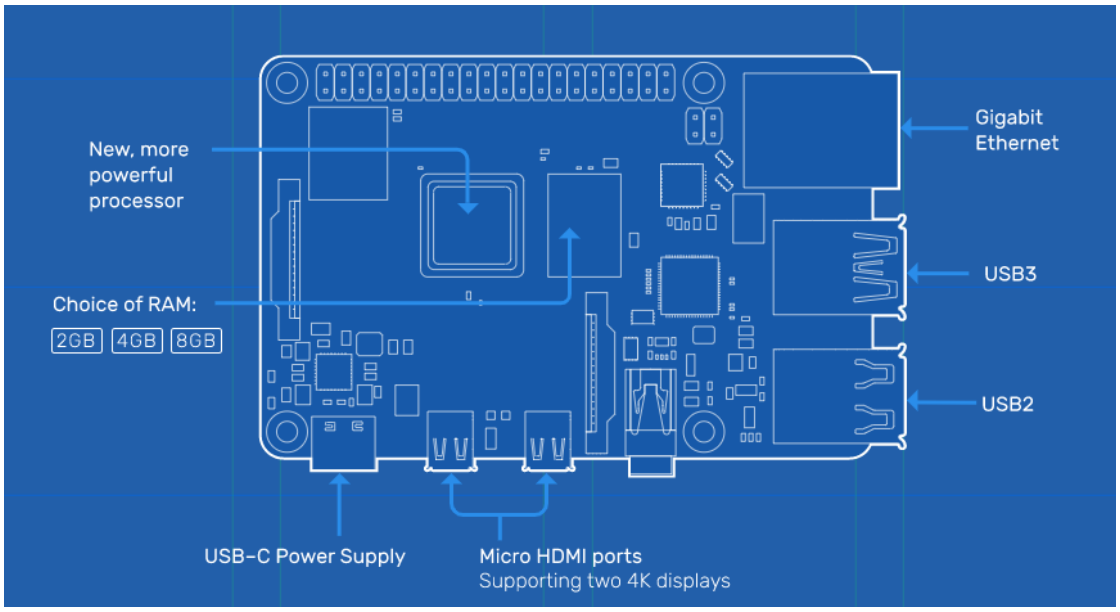


Figure 1- Raspberry Pi 4 overview courtesy to raspberry pi for image

2.2 Raspberry Pi High Quality Camera

This Camera has a Sony IMX477R back illuminated sensor. This sensor is 12.3 megapixel and is connected using a ribbon cable. Since this camera sensor doesn't come with a lens, a suitable one was bought for this. It is a 6mm wide lens that is suitable for CCTV use.[13]

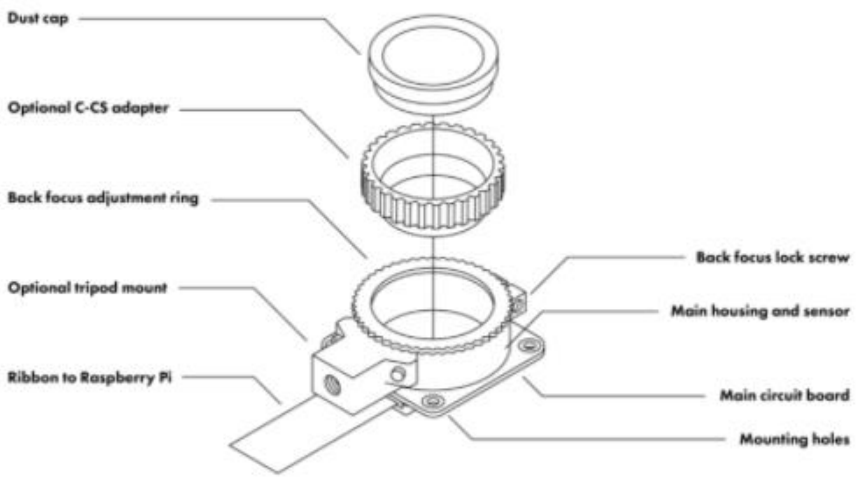


Figure 2- exploded Diagram courtesy to Raspberry Pi for Image

Chapter 3: Software tools

3.1 Python/ OpenCV

This projects uses Python language together with OpenCV video processing module for controlling the camera. This module allowed for the recording of the video. This module is capable of much more such as:

- Frame Analysis
- Facial Recognition
- Object Recognition
- Video/ Image editing.

While not all OpenCV Utilities were utilised in this project, Certain functions could in the future be Incorporated to the project. For example, OpenCV could be used to detect cars driving in the driveway of a house so that a notification could be sent to the user that a car is approaching.

Using this module, a motion detection feature was able to be implemented. This allowed to save precious memory. [6]

3.2 HTML/ PHP

As I will discuss in this report, a website will be used to watch videos. HTML will be used as user input so they can find the video they want to view. The information is entered into a html form and is analysed using php.

It should also be noted that the html method POST this is for security reasons, so user information isn't displayed on the URL or history

3.3 Crontab

Crontab is also used in this project as a way of scheduling different code/ files. In this project it is used in two different ways. The first way is to record the videos daily, the second way is to use it to delete older files . These different functions will be discussed in Chapter 6 and Chapter 9. To further understand the how Crontab is used, a website was used. [11]

Chapter 4: System Installation

4.1 Camera

For this project, the camera is central to the project, as it was important that the quality of the video be as good as possible. The Raspberry Pi 4 model also has an I/O port specifically for the camera which allows the camera to connect directly to on-board GPU which saves some much-needed processing power for the CPU. There we three cameras which were considered for this for this project. The data sheet was considered below.

	Camera Module v1	Camera Module v2	HQ Camera
Net Price	$25	$25	$50
Still Resolution	5 Megapixels	8 Megapixels	12.3 Megapixels
Video Modes	1080p@30, 720@60 & 640 *480@60/90	1080p@30, 720@60 & 640 *480@60/90	1080p@30, 720@60 & 640 *480@60/90
Sensor	OmniVision OV5647	Sony IMX219	Sony IMX477
Integrated Lens	yes	yes	No
Year of release	2013	2016	2020

After Reviewing the information within the spec sheet, while it is the most expensive. Having the ability to change the lens to suit the environment, is a very good advantage. This is one of the advantages of using the High-Quality camera. This camera also provides the highest resolution.[15]

4.2 Raspberry Pi Operating System

For this project, a 64-bit Ubuntu operating system was needed. The reason why it needs to be used is because, for this project a python module named 'OpenCV' was used. Initially in the project, OpenCV wouldn't install properly, and the problem was found to be due to the Default Raspbian only being 32-bit. To remedy this the Raspberry Pi distribution of 64-bit Ubuntu was used. To install the ubuntu system, first the image file of the OS should be downloaded onto a laptop along with the Raspberry Pi imager. Once both are downloaded, the Raspberry Pi imager was used to flash the image onto a micro-USB. The OS is then ready to boot up once this is done.

4.3 Installing development Languages

For this project, Python, PHP,HTML,CSS was used in order to develop the camera monitoring system. Apart from HTML and CSS, these languages and module needed to be installed onto the Raspberry Pi. The following commands were used to install the necessary languages for the project:

```
Sudo apt-get python3
```

```
Sudo apt-get python3-pip
```

```
Sudo apt-get php
```

4.4 Third party Code/ Database

For this project, MariaDB, OpenCV and apache2 webserver software will be used in this project.

4.4.1 OpenCV

The installation for OpenCV required the use for python pip. This means that we need to use the following command.

```
Pip install opencv-python
```

This part of the project turned out to be one of the big problems of the project. This is because when starting the project, the Standard Raspberry Pi 32-bit system Raspbian was used which was incompatible with the version of OpenCV needed. It was called Raspbian and is the OS made by Raspberry Pi. To solve this problem, a 64-bit ubuntu OS was used. Without using a 64-bit OS this module would not be able to be used for this project.

4.4.2 MariaDB

The MariaDB database was used in this project, the installation for this required more set up. In the Ubuntu CLI the following command was used to install maria DB

```
Sudo apt install mariadb-server
```

Once it is installed, a user needs to be setup for the database, so that when creating the login form for them on the html page. This is done by connecting to the database and then creating the user and giving it appropriate permissions. Please note the commands below must be in the order shown.

```
Sudo mysql

CREATE USER  ‘user’ IDENTIFIED BY ‘password’

GRANT ALL PRIVILEGES ON TO ‘USER’

FLUSH PRIVILEGES
```

It is important to note that the user were given all privileges as there were very few security risks. Although if there are security risks associated in the environment, more measures can be taken to ensure the security and safety of accounts. For example, there is a security package that can be installed with MariaDB. The command: *sudo my_sql_secure_installations* this can-do multiple things to secure the database such as Disallowing the root user to login remotely.

Another Security feature which could be added is PHP filters, these filter test the inputs given by the user. These checks help to make sure that any information isn’t malicious and wont damage the integrity of the database or take information from it [16]

4.4.3 Apache2

This was installed using the following command in the Linux CLI, we also need to change some of the setting for the File path for the Document Root. This is done by first going to the going to the right directory using the following command*: cd /etc/apache2/sites-available*. The file named *000-default.conf* is unwritable so to change that we need to use the command *chmod 777 000-default.conf*. this will make the file writable, executable, and readable to all users. the file was then edited with the command *sudo nano 000-default.conf* and change the document root to the file path that was used for this project. After the file has been edited, the apache2 service needs to be restarted for the changes to go into effect. This can be done using the command *sudo service apache2 restart*.

Chapter 5: Design of webpage- Wireframe Diagram

5.1 Login Page

As the login page is the first page that the user is greeted with, the aim for this page is to set the theme for the entire website. For this project the Login page will be a minimalist design with darker colours. Refer to Figure 3 to see the overall layout for the page.

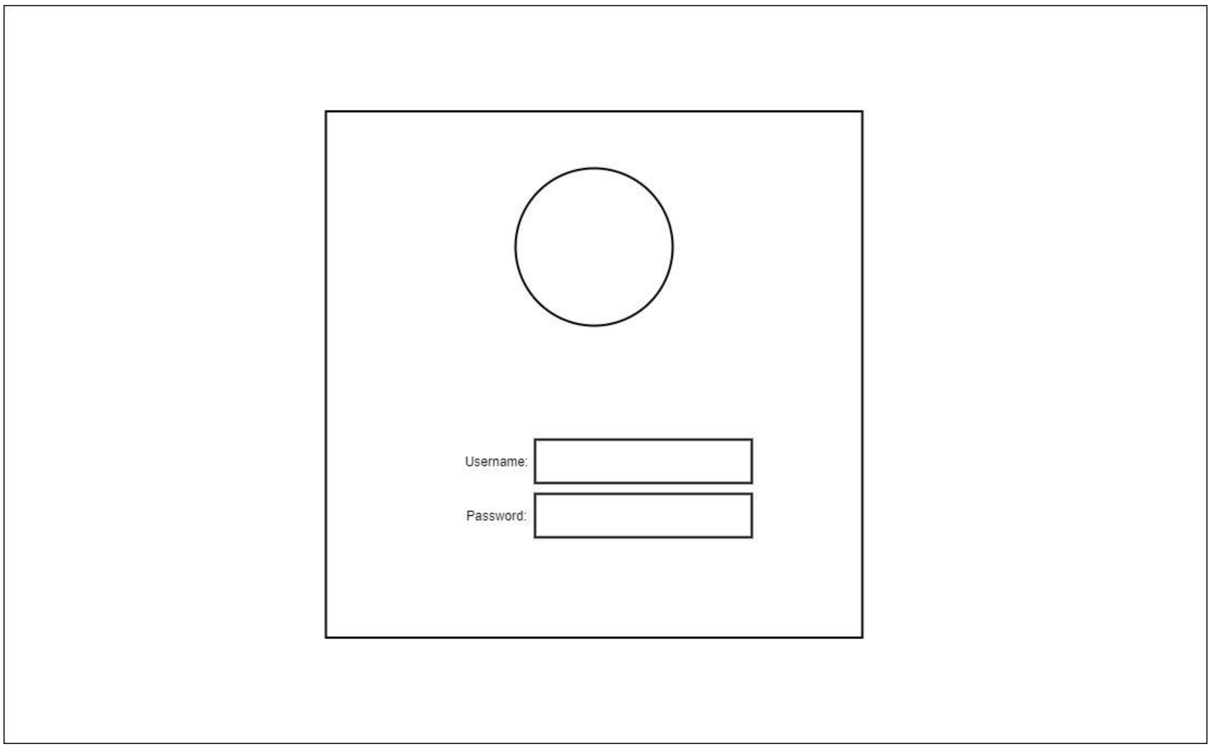


Figure 3

5.2 Page for User Input

For this page I wanted to also make it minimal with design. The idea for this page is to provide a space for the user to search for what they want to view. It is also important that the top navigation is present, not only so if the wrong video is select but also to allow the user to log out and destroy the session. Refer to Figure 4 for the overall layout of the page.

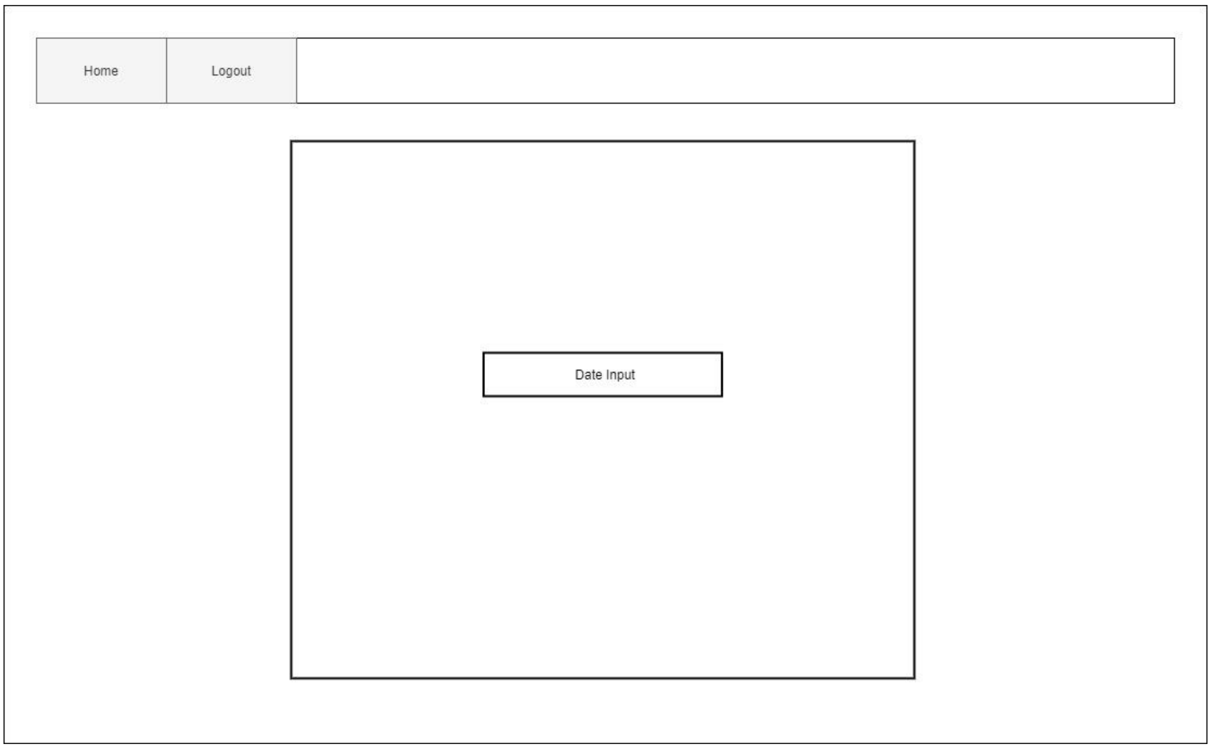


Figure 4

5.3 Page for Displaying Video

This page only has one function which is to display the video. To avoid confusion, the date of the video which is playing also is displayed for the user. It is also important that the navigation bar is also displayed on this page, so the user has the ability to get back to the previous page to find a new video. Refer to Figure 5 for the layout of the page.

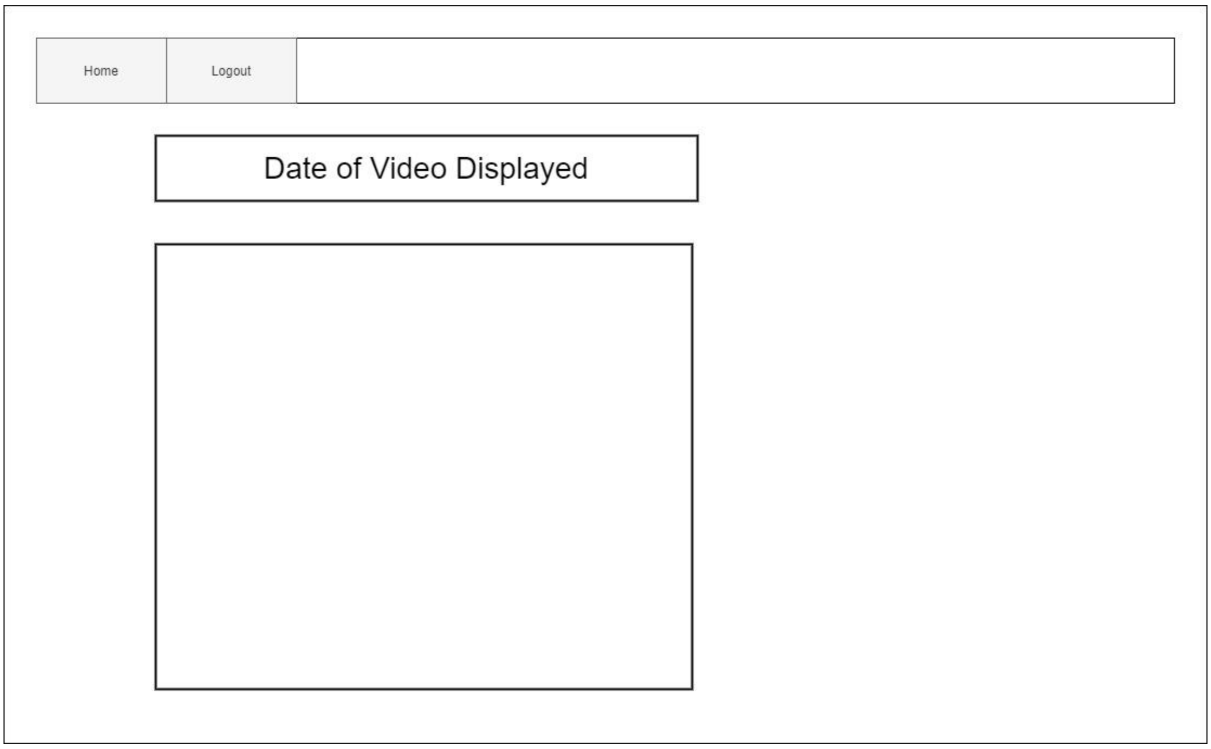


Figure 5

Chapter 6: Recording Video

6.1 Video Codec

Before completing the code, it was important to decide the codec which was used to save the video. First the file format was decided. There are 3 file formats that work by default with a html page, they are MP4, OGG and WEBM. However, MP4 is the only that work on the majority of web browsers.

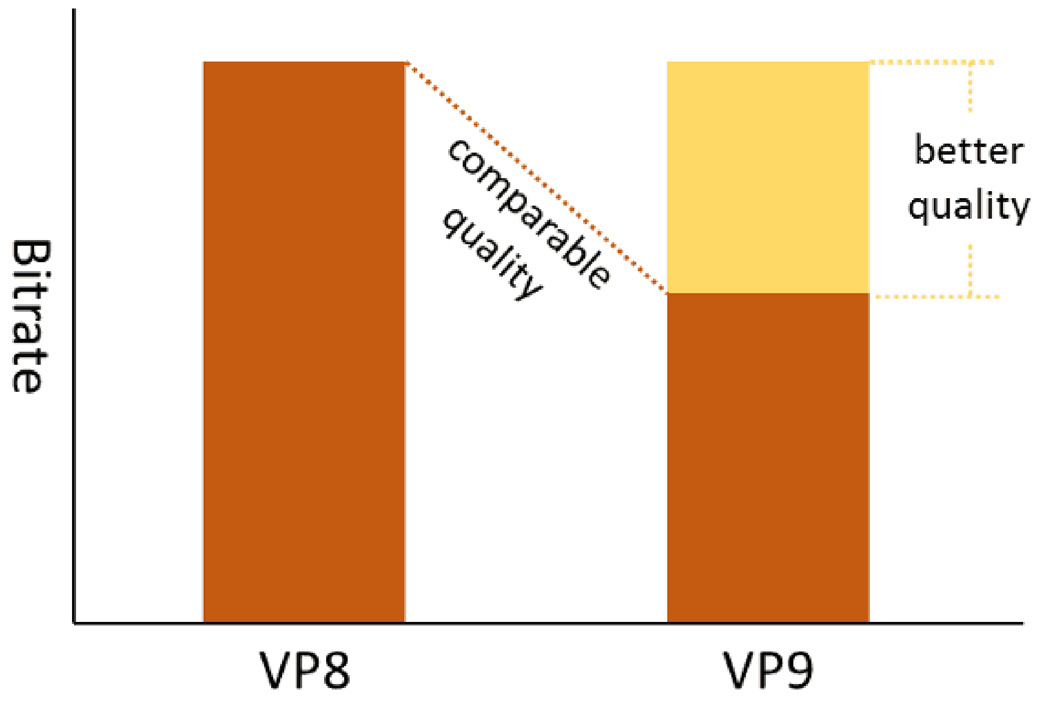
Finding a suitable codec for this project turned out to be a difficult part of the project.

6.1.1 MP4

if MP4 format was chosen, a suitable codec would need to be chosen such as H.264 or Motion JPG. It should be noted that if the video is to be displayed on a html website, specific codecs are necessary this is why h.264 codec is recommended for using the MP4.

6.1.2 OGG

If the user wishes to use a different video format such as the OGG, it is recommended that we use a different Codec such as Theora or Vorbis. If the Vorbis is chosen, it is recommended VP9 is used since VP9 is better than VP8 as it is newer. It is also able to provide much more higher quality compressed video. However, VP8 takes up less resources than VP9, so it depends on the situation for which is used.[17]



6.1.3 WEBM

Similarly, to OGG format, the Theora/ Vorbis codecs allow us to save the videos

6.2 Implementing Codec

As seen in Appendices Recording Video, in this code, a four-character code is needed. For example, the four-character code for H.264 is ‘avc1’. Below is a list of different codecs with their codes.

Codec	Code	Used For
H.264	Avc1	MP4
Motion JPG	Mjpg	MP4
Vorbis	VP08	OGG/WEBM

6.3 Framerate

When considering the framerate, it was important to consider the fact that if too many frames are recorded, too much memory will be used up for the recording. However, if not enough frames are recorded the recording will look choppy. I found that between 20 and 25 frames provide the minimum number of frames.

6.4 Crontab

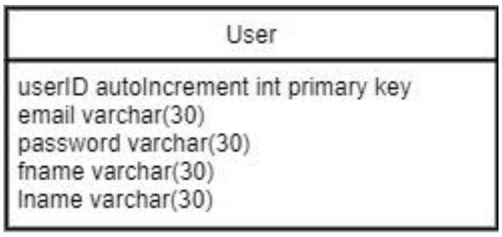
As seen in the code, the code records for a certain length of time and then ends. This was done on purpose so that I could implement crontab. Crontab is a scheduling programme which allows the user to decide when a programme should be executed.

Chapter 7: Displaying Video

For this project, there were different avenues for viewing the recordings. For example, the recordings could be sent to a storage solution, or access the raspberry pi using a remote access solution. However due to the nature of the project, it was considered that a website was the best solution. As discussed in the Installation chapter, we are using the apache2 webserver, as this allows us to look up the ip address of the Raspberry Pi, when the website is displayed.

7.1 Account Security

As Discussed, the project is using a login system which stores user information in the MariaDB Refer to the diagram below to understand the layout of Database. While no foreign key tables were necessary for this project. A primary key was made in order to make sure that if the database needed to be expanded, they could be without much trouble.



User
userID autoIncrement int primary key email varchar(30) password varchar(30) fname varchar(30) lname varchar(30)

It should also be noted that the form information is being sent using the POST method. The reason why this method is used over one like GET is because, while GET can be useful for certain situations, it lacks a lot of security/ privacy. For example, if GET was used the user information would be shown in the browser/ history.

7.2 Website

When accessing the videos to view, having all videos on one page felt overwhelming. To solve the website was developed to have a form so the user can search the date of the video they want to look at. When designing the website, I also wanted the loop for the website to make sense, so when a user look up a date instead of just pressing the "Back" button on a browser. I wanted to give the user a way to get back to where the user can search. So, a navigation bar was developed so the user can easily navigate the website[8]

7.3 Sessions

For the benefit of the users, sessions were added to the website so that, if the users accidently closed the website, they could simply go back, and they wouldn't have to log in again.

7.4 Code Explanation

7.4.1 User Login Form

While most of the html code is about designing, so the forms the following will be what is explained.

```
<div style="padding: 15px 32px;margin: auto;background-
color:#161618f5;width:400px;length:200px;border-
radius:25px;top:15;float:center;"align=center>
        <form action="CheckLogin.php" method="post">
            <div style="text-align: center;margin: 24px 0 12px 0;">
                <img src="viking.png"style="width: 40%;border-
radius: 50%;background-color:#46464ef5;;">
            </div>

            <h1 style="color: grey;text-align: center;">User Login</h1>
            <label for="email" style="color: grey">Email: </label>
            <input type="text" id="email" name="email"><br><br>
            <label for="pass" style="color: grey">Password: </label>
            <input type="password" id="pass" name="pass"><br><br>
            <input type="submit" value="submit">
        </form>

    </div>
```

Figure 6

As you can see from Figure 6 above, this is a form which the user would fill in to sign into their account, the method used for passing the data is by POST which is more secure than other html methods like GET. This data is then passed to a php page which then checks the user input values.

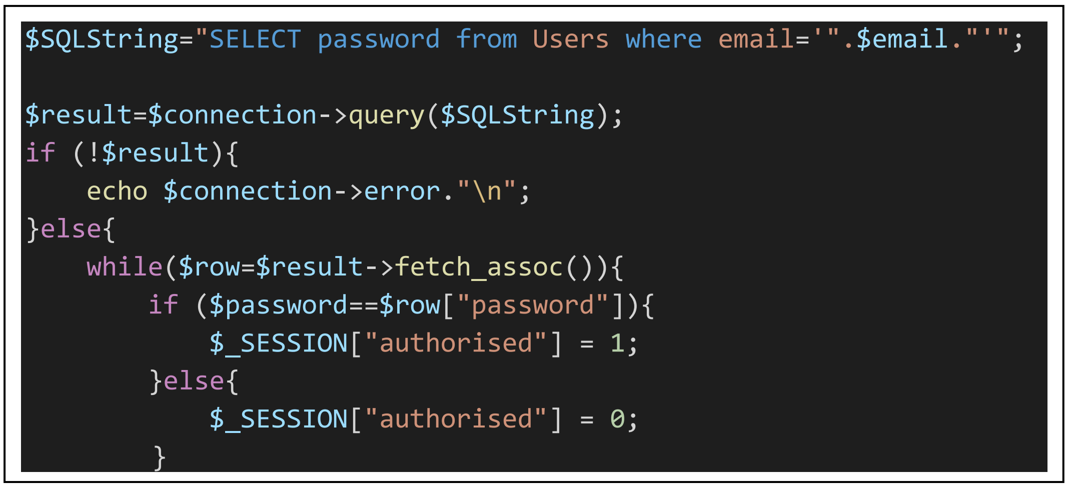
7.4.2 Check Login

To see the full code, refer to the Check Login Section in the appendix A

```
$servername="localhost";
$dbusername="user";
$dbpassword="password";
$dbname="OdinsEye";
```

Figure 7

To begin with, as discussed in the Installation section for MariaDB, an account needs to be set up. This is so that when information that the user inputs, this php file can connect to the database to check the relevant data. To see how to connect to the MariaDB database, use the code in Figure &.

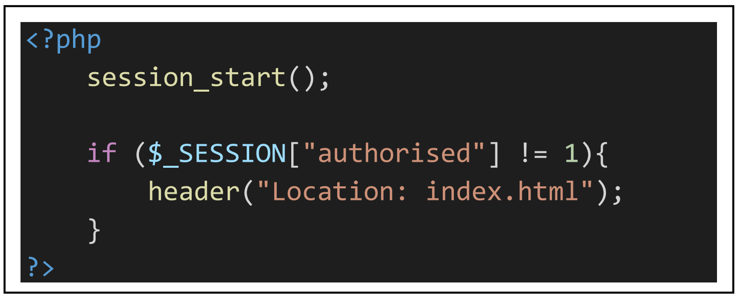


```
$SQLString="SELECT password from Users where email='".$email."'";

$result=$connection->query($SQLString);
if (!$result){
    echo $connection->error."\n";
}else{
    while($row=$result->fetch_assoc()){
        if ($password==$row["password"]){
            $_SESSION["authorised"] = 1;
        }else{
            $_SESSION["authorised"] = 0;
        }
```

Figure 8

Figure 8 is the SQL query is made, the query selects all that has the email of the one given. After that the password is checked and if the password is right the session begins



```
<?php
    session_start();

    if ($_SESSION["authorised"] != 1){
        header("Location: index.html");
    }
?>
```

Figure 9

It should be noticed that every page of the website starts with the code as seen in Figure 9, so if the user has no session, they are redirected to the login page. Refer to Figure 9 to see how Sessions are managed for each webpage.

Chapter 8: Ethical/ Commercial/ Safety/ Environmental

This project is mostly a software development project. The risks are summarised on the completed project safety statement. That being said, due to the fact that this project does relate to surveillance, the following factors need to be discussed.

8.1 Privacy

Considering the fact that this project is developed as a surveillance tool, privacy is a key ethical issue of the project. The issue with the use of camera surveillance is becoming more and more of an issue in today's society, with companies like Google promising to adhere to strict standards of GDPR. These laws are regularly broken or usurped through terms and conditions, and user's data is sold. It was for this reason that the project was developed so that people could control their own data.

8.2 Security

Security is always an issue when it comes to database and account privacy. The reason why security was something that needed to be address was because if there was no security, anyone could view the user's recordings. To counter this, a login check was created using SQL select statements retrieve account emails and passwords.

Another issue that arises is SQL injection attacks. To counter this, When the login details is being checked it uses filters to ensure that the inputted information is not an SQL injection attack

8.3 Commercial

While Developing this project, there was no concern about privacy. However, if the project ever gets into general usage. There is some concern as to the use of the camera system. In order to use a camera system in a domestic setting, the user must ensure that only property of the user is recorded, anything else would be subject to action taken by the Data Protection Commission.

Chapter 9: Video Storage management

As discussed, storing video can become very cumbersome if not managed. To solve this issue, a few solutions were researched for this project. The first one was to use a cloud storage solution, the second was to incorporate motion detection into the recording of videos, as within a full day a lot of wasted video would be recorded. Motion Detection would reduce video size by only recording scenes contain movement. The third option would be to use a motion detecting sensor so similarly to the second option, when motion is detected, the recording python file starts to play.

9.1 Cloud Storage Solution

When considering this option, it was important consider the objective of the project which was to provide a subscription-less service of the google nest or amazon ring camera. With the amount of data that would need to be stored, a paid solution would be needed for cloud backup, which might be ok for some users but would however be useful as an option for others.

An advantage of using this storage solution is that if this website were to become public so the user could view clips from where ever they are, having an online storage solution would greatly benefit the development of the new website as storage on a specific storage solution like an amazon bucket, is a lot cheaper than storing the videos on the instance where the website is stored.

Overall, for the expansion of the website to allow public access, this solution would be the best.

9.2 Motion Detection Software Solution

Using OpenCV allows for Motion Detection through the use of video frame analysis. This is done by storing a background frame to be used as a reference for what the environment looks like. During video monitoring each new frame would be compared to this reference frame and recording is started only when there are significant differences. This is a complex process and due to time constraints only a rudimentary motion detection feature was developed.

This method also keeps the video shorter while all the important events are recorded. Which means that potentially much less data is needed to store video files on the Raspberry Pi

9.3 Motion Detecting using a movement Sensor

Using this method would involve using a Motion Detecting sensor in order to sense when there's movement. The advantage of this is that this method would use less processing power than the image processing approach because I/O port would have to be checked in order to know if the video should be recorded. Similarly, to the software motion detect, this method saves on data storage.

Another advantage of using this approach is that if the environment changes. The sensor will still operate, and the recording will stop since there is no motion with the software solution, if a change is made to the environment after the reference frame is made each new frame would constantly be recognised as a change, unless the reference frame was updated

9.4 Motion detecting using video processing

As can see from in Appendix B- Recording Video, each frame needs to be pre-processed in order to get reliable measurement for differences in each frame. In this section different parts of the code will be explained as to why it is use. The pre-processing involved three steps:

1. Each frame is resized to a fixed size to avoid variability introduced from cameras with different image resolution. This step also ensures that the resized frame has the same aspect ratio as the camera frame to reduce image distortion.
2. The resized colour frame is converted to grey scale to reduce processing time.
3. The greyscale frame is filtered to reduce noise and image compression artifacts.

In this following section different part of the code are explained.

9.4.1 Reference Frame

```
aspect_ratio = frame_width/frame_height

# Dimensions for resized motion detect frames
md_frame_width = 512
md_frame_height = int(md_frame_width/aspect_ratio)
md_size = (md_frame_width, md_frame_height)
num_md_pixels = md_frame_width*md_frame_height
```

This section of code resizes the frame of the image. The aspect ratio of the camera frame is computed to ensure that the resized image is not distorted. In the above code snippet, the frame width is set to 512 pixels and the width defined by the camera aspect ratio. It is also good to resize the image to be smaller to reduce the processing as motion detection does not require high resolution. The number of pixels in the resized frame, *num_md_pixels*, is used later in the motion detection code.

```
startTime = time.time()

while(int(time.time() - startTime) < 60):
    ret, frame = video.read()
    # Normalise frames by resizing, converting to monochrome
    # and smoothing to reduce noise and compression artifacts
    if ref_frame is None:
        ref_frame = cv2.resize(frame, md_size)
        ref_frame = cv2.cvtColor(ref_frame, cv2.COLOR_BGR2GRAY)
        ref_frame = cv2.GaussianBlur(ref_frame, (21,21), 0)
        ref_level = np.sum(ref_frame) / num_md_pixels
```

For this part, the current time is stored in the variable *startTime*. This is to help with the time keeping aspect of the project. As is seen in the while loop, the current time is being subtracted by the Start Time and being converted into an integer. This then allows for the while loop to time out, for testing purposes I simply used 60 seconds. This could be increase to a longer period of time in a real application.

When the programme is starting the reference frame is none, so the first frame is taken as the template for what the environment should look like. So, the frame is resized, and colour scale is changed from the RGB scale to the Grey scale. This is done so that is colours are changed due to environmental factors such as sunlight, the difference won't be noticed by programme. A Gaussian Blur smoothing filter is then applied to reduce noise. The average brightness of the frame is calculated and stored in variable ref_level. This information is used to compensate for changes in the environment background light level as a simple threshold test is used detect motion. [1]

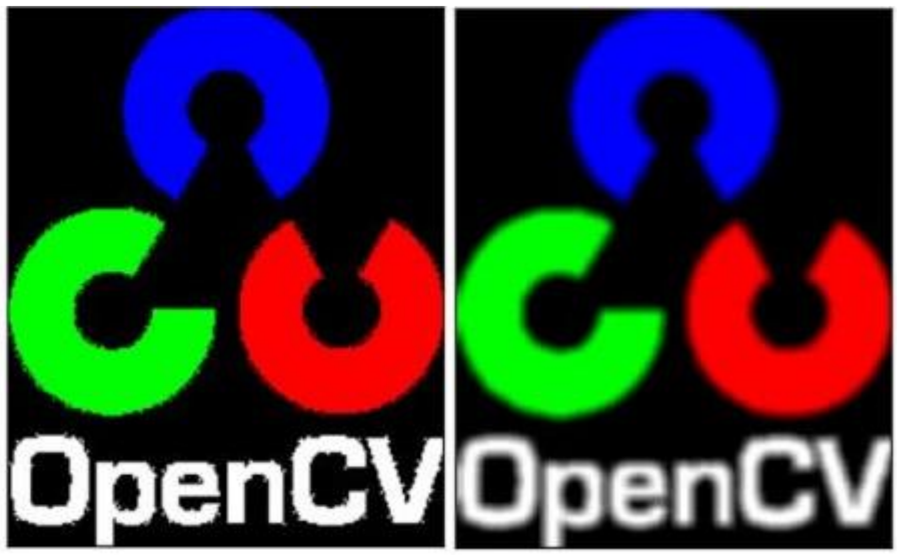


Figure 10- courtesy to OpenCV for image

As seen in Figure 10 the Gaussian Blur is show. The removes the noises off the edges of obje

9.4.2 Current Frame

After the first frame is saved as a reference, the process is repeated for new frames, but this time it is compared against the reference.

```
else:
        md_frame = cv2.resize(frame, md_size)
        md_frame = cv2.cvtColor(md_frame, cv2.COLOR_BGR2GRAY)
        md_frame = cv2.GaussianBlur(md_frame, (21,21), 0)
        md_level = np.sum(md_frame) / num_md_pixels
        #Implement brightness compensation
        alpha = ref_level/md_level
        md_frame = cv2.convertScaleAbs(md_frame, alpha, beta = 0)

        #Get difference between reference and current frames
        frameDelta = cv2.absdiff(ref_frame, md_frame)
```

the variable alpha stores the relative brightness between reference and current frame. It is used by the OpenCV convertScaleABS function to multiply the current frame pixel levels by alpha so the average brightness equals the reference frame average brightness. The last line of code uses the absdiff function to compute a new delta frame whose pixels corresponds to the absolute difference between reference and current frame.

```
#Alternatively get sum of pixels above trigger level
        dif_level = np.sum(frameDelta[frameDelta>trig_level])
        average_dif_level = dif_level/num_md_pixels
```

This code adds up all the pixels of the delta frame that are above a specified trigger level stored in variable *trig_level*. This ensures that small differences between the frames are ignored. The value for the trigger level is found by trial and error measurement and a value of 10 pixel levels was chosen from tests. After that the dif_levelvalue is average by dividing the number of pixels in the analysis frames.

This compensates should the analysis frame size be changed in future as the *average_dif_level* would remain relatively unchanged.

```
        if average_dif_level >= 50:
            frame = cv2.putText(frame,str(datetime.datetime.now()),(10, 450),c
v2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),1,cv2.LINE_8)
            result.write(frame)
            cv2.imshow('Frame', frame)

            # Press S on keyboard
            # to stop the process
            if cv2.waitKey(1) & 0xFF == ord('s'):
                break


video.release()
result.release()

# Closes all the frames
cv2.destroyAllWindows()

print("The video was successfully saved")
```

The decision on whether motion is detected is done by the if statement which compares the average difference in pixel levels to a threshold value of 50. This number is obtained by test and will likely change. Due to time constraint more testing is needed for it to be evaluated properly.

after all the checks are done, and if the motion detection is triggred, the frames are saved and a timestamp is added to the frame. For testing purposes, a way to stop recording was made. If when recording the video the user wants to stop before the time of recording is over, the user can press the s key to stop the video.

```
if (!filter_var($email, FILTER_SANITIZE_STRING) === false || !filter_var($pass
word, FILTER_SANITIZE_STRING) === false){
    header("Location: index.html");
    echo "Enter a valid user information";
}
```

The above if statement, is used to filter any user input which may aim to attack the integrity of the Database. An example of this would be SQL injection attacks. While there are many different filters which could be implemented, the filter which sort strings was the one which was used for this project.

9.5 Crontab

As discussed, crontab is also used to delete videos. This is done to ensure that if older videos aren't being used than it can be safely deleted after a set amount of time. For example, if a video is a week old, then it should be deleted so that there is space for the new videos .

Chapter 10: Conclusion

In conclusion, while some difficulties were anticipated when starting this project. such as the managing storage space, and some difficulties were found along the way such as the issue with finding the correct codec to use with the HTML website. The solutions to these problems provide an alternative solution to using camera services provided by big tech companies. While providing the user the ability to maintain their own data.

Also, while more features could be added to this project to develop the project further. The features provided already, provide the needed proof of concept for the project

11.1 Potential Improvements

While The project does fulfil the aim that was set out. There are a few potential improvements that could be made to the project. these are as follow.

11.1.1 Notification service

For this project, OpenCV is being used to implement motion detection. This is done to save memory of the raspberry pi. A new feature could also be implemented to notify the user as to when there is an event. This could be done a number of ways such as by email or by SMS message

11.1.1.1 Email

Sending email would seem to be the easiest course of action to accomplish this task. This is because python has an inbuilt module call smtlib. An advantage of using this is that its free and stable since the raspberry pi is connected through ethernet. Another Advantage is that smaller clips of events could be sent by email for free.[2]

11.1.1.2 SMS

To use SMS as a form of notification, a library such as Twilio could be used. This version of notification could be argued that it is better since everyone has phones, but not every has the ability to check emails on them. Although that argument is becoming less and less accurate. That being said one big disadvantage of using Twilio is that SMS cost money to use.[3]

11.1.1.3 Overall Preference

While different people have different opinions on which would be better. For this project, email should be the one that is used. The reason why email works better is because of the ability to send smaller clips. Most people are also able to access emails from their phones as well.

11.1.2 Public Domain Website

As discussed earlier the possibility of using a public accessed website was one considered so the user could view videos their recordings from anywhere. While it wasn't utilised for this project, it is a potential improvement to the project. If this were to be added, some significant changes to the project would need to take place.

11.1.2.1 Amazon S3 cloud storage

As Discussed in the storage section, if a public Domain website is created using an online storage would be very useful for the project. Amazon provide a storage solution called an AWS S3 Bucket. Using this as my storage solution would allow me to like it to an Amazon EC2 server using Amazon IAM. While these buckets are able to be used to host websites, for this project I need access to languages like PHP. Unfortunately, Amazon Buckets are only able to host static websites.[14]

11.1.2.2 Amazon EC2 Instance

Amazon Provide Server Space, they call this Service EC2 instance. Amazon provide access to all Operating systems, but for this project a Linux based EC2 instance. As discussed, the point of this server is just for the website, as if the videos were to be stored on this server it would cost too much to maintain. As Discuss the reason why an EC2 Instance is used is so that I can develop a Dynamic Website using PHP.

11.1.2.3 MariaDB

In order to give access to multiple users from anywhere. The database would have to be reorganised so that a user logging in, will only be able to see videos relating to them. This problem can be solved by setting up an additional column, this column would function as the name of a given directory. Once this string reviewed using a SELECT SQL statement, it could then be used to access the directory for that person on the system.

11.1.3 Image Recognition

OpenCV image processing code could also be developed further by implement a recognition feature for if there are people or pets coming through an environment that would normally be in that environment, the System would recognise that they are supposed to be there. However, if an unrecognized person or arrives, then the system would then record.

The use of a servo motor could also be implemented so that if there is a disturbance, the camera centres on the disturbance. The motion detection that is currently used would have to be reworked if this feature were to be used. This is because if the reference image is saved, and then the camera were to move, then the camera would constantly be recording.

11.1.4 Livestream

Another improvement that could be made to the website, is to have a livestream of the camera. This would provide a feature that would work similarly to the one found on next where users could view the camera feed. This would work well with the notification system because if an email notification system was in place. A link could be provided in the email so the user could easily access the livestream.

11.1.5 DNS Server

Making a DNS server would be good for user experience. As it stands, the user has to look up the IP address of the Raspberry Pi. If a DNS server were to be created, the user would have to put in a normal URL and then the DNS server would find the appropriate IP address

Bibliography

[1]"OpenCV: Smoothing Images", Ccoderun.ca, 2021. [Online]. Available: https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html. [Accessed: 12- May- 2021].

[2]"email — An email and MIME handling package — Python 3.9.5 documentation", *Docs.python.org*, 2021. [Online]. Available: https://docs.python.org/3/library/email.html. [Accessed: 12- May- 2021].

[3]"Build a SMS Chatbot With Python, Flask and Twilio", *Twilio Blog*, 2021. [Online]. Available: https://www.twilio.com/blog/build-a-sms-chatbot-with-python-flask-and-twilio. [Accessed: 12- May- 2021].

[4]R. David, "Chapter 10: Movement Detection With Background - Robin David", *Robindavid.fr*, 2021. [Online]. Available: http://www.robindavid.fr/opencv-tutorial/chapter10-movement-detection-with-background.html. [Accessed: 12- May- 2021].

[5]A. Rosebrock, "Basic motion detection and tracking with Python and OpenCV - PyImageSearch", *PyImageSearch*, 2021. [Online]. Available: https://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/. [Accessed: 12- May- 2021].

[6]"OpenCV: OpenCV Tutorials", *Docs.opencv.org*, 2021. [Online]. Available: https://docs.opencv.org/master/d9/df8/tutorial_root.html. [Accessed: 12- May- 2021].

[7]"Saving Operated Video from a webcam using OpenCV - GeeksforGeeks", *GeeksforGeeks*, 2021. [Online]. Available: https://www.geeksforgeeks.org/saving-operated-video-from-a-webcam-using-opencv/. [Accessed: 12- May- 2021].

[8]"CSS Navigation Bar", *W3schools.com*, 2021. [Online]. Available: https://www.w3schools.com/css/css_navbar.asp. [Accessed: 12- May- 2021].

[9]A. Rao, "OpenCV Python Tutorial | Computer Vision Using OpenCV | Edureka", *Edureka*, 2021. [Online]. Available: https://www.edureka.co/blog/python-opencv-tutorial/. [Accessed: 12- May- 2021].

[10]*Viking*. https://www.flaticon.com/free-icon/viking_1396724?term=viking&page=1&position=18&page=1&position=18&related_id=1396724&origin=search, 2021

[11]"Crontab.guru - The cron schedule expression editor", Crontab.guru, 2021. [Online]. Available: https://crontab.guru/#5. [Accessed: 13- May- 2021].

[12]"Raspberry Pi 4B specs", Raspberrypi.org, 2021. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/. [Accessed: 13- May- 2021].

[13]"Raspberry Pi HQ Quality", Raspberrypi.org, 2021. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-high-quality-camera/. [Accessed: 13- May- 2021].

[14]"Buckets overview - Amazon Simple Storage Service", Docs.aws.amazon.com, 2021. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingBucket.html#about-access-permissions-create-bucket. [Accessed: 13- May- 2021].

[15]"Camera Module - Raspberry Pi Documentation", Raspberrypi.org, 2021. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/camera/README.md. [Accessed: 13- May- 2021].

[16]"PHP Filter Functions", W3schools.com, 2021. [Online]. Available: https://www.w3schools.com/php/php_ref_filter.asp. [Accessed: 13- May- 2021].

[17]"VP8 vs VP9 - Is this about Quality or Bitrate? • BlogGeek.me", BlogGeek.me, 2021. [Online]. Available: https://bloggeek.me/vp8-vs-vp9-quality-or-bitrate/. [Accessed: 13- May- 2021].

Appendix A- Checking Login

```php
<?php
$email=$_POST["email"];
$password=$_POST["pass"];

$servername="localhost";
$dbusername="user";
$dbpassword="password";
$dbname="OdinsEye";

if (!filter_var($email, FILTER_SANITIZE_STRING) === false || !filter_var($pass
word, FILTER_SANITIZE_STRING) === false){
    header("Location: index.html");
    echo "Enter a valid user information";
}

session_start();

$_SESSION["authorised"] = 0;

$connection= new mysqli($servername,$dbusername,$dbpassword,$dbname);

if ($connection->connect_error) {
    die("Failed to connect to database\n".$connection.connect_error);
}

$SQLString="SELECT password from Users where email='".$email."'";

$result=$connection->query($SQLString);
if (!$result){
    echo $connection->error."\n";
}else{
    while($row=$result->fetch_assoc()){
        if ($password==$row["password"]){
            $_SESSION["authorised"] = 1;
        }else{
            $_SESSION["authorised"] = 0;
        }
    }
}

if ($_SESSION["authorised"] == 1){
    header("Location: PickVideo.php");
}else{
    header("Location: index.html");
}
?>
```

Appendix B- Recording Video

```
import cv2
import numpy as np
from datetime import date
import datetime
import time
import sys


trig_level = 10
firstFrame = None
filename = date.today().strftime("vids/%d-%m-%Y")+".mp4"
print(date.today().strftime("%d-%m-%Y"))

video = cv2.VideoCapture(0)

if (video.isOpened() == False):
    #print("Error reading video file")
    sys.exit("Error reading video device")

frame_width = int(video.get(3))
frame_height = int(video.get(4))
size = (frame_width, frame_height)
aspect_ratio = frame_width/frame_height

# Dimensions for resized motion detect frames
md_frame_width = 512
md_frame_height = int(md_frame_width/aspect_ratio)
md_size = (md_frame_width, md_frame_height)
num_md_pixels = md_frame_width*md_frame_height


result = cv2.VideoWriter(filename, cv2.VideoWriter_fourcc(*'a','v','c','1'), 2
4, size)

ref_frame = None    #Motion detect reference frame
startTime = time.time()

while(int(time.time() - startTime) < 60):
    ret, frame = video.read()
    # Normalise frames by resizing, converting to monochrome
    # and smoothing to reduce noise and compression artifacts
    if ref_frame is None:
        ref_frame = cv2.resize(frame, md_size)
        ref_frame = cv2.cvtColor(ref_frame, cv2.COLOR_BGR2GRAY)
        ref_frame = cv2.GaussianBlur(ref_frame, (21,21), 0)
        ref_level = np.sum(ref_frame) / num_md_pixels
    else:
        md_frame = cv2.resize(frame, md_size)
        md_frame = cv2.cvtColor(md_frame, cv2.COLOR_BGR2GRAY)
        md_frame = cv2.GaussianBlur(md_frame, (21,21), 0)
        md_level = np.sum(md_frame) / num_md_pixels
        #Implement brightness compensation
        alpha = ref_level/md_level
        md_frame = cv2.convertScaleAbs(md_frame, alpha, beta = 0)

        #Get difference between reference and current frames
        frameDelta = cv2.absdiff(ref_frame, md_frame)

        #Count number of pixels above a defined trigger level
        #num_dif_pix = np.sum(frameDelta>trig_level)

        #Alternatively get sum of pixels above trigger level
        dif_level = np.sum(frameDelta[frameDelta>trig_level])
        average_dif_level = dif_level/num_md_pixels

        print(average_dif_level)


        if average_dif_level >= 50:
            frame = cv2.putText(frame,str(datetime.datetime.now()),(10, 450),c
v2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),1,cv2.LINE_8)
            result.write(frame)
            cv2.imshow('Frame', frame)

            # Press S on keyboard
            # to stop the process
            if cv2.waitKey(1) & 0xFF == ord('s'):
                break


video.release()
result.release()

# Closes all the frames
cv2.destroyAllWindows()

print("The video was successfully saved")
```